MASTER EN SCIENCES INFORMATIQUES

Didacticiel d'Aide à l'Analyse et à la Traduction de Textes Latins

Jacob, Sophie

Award date:
1988

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Faculté d'Informatique
Année Académique 1987-1988

"Didacticiel d'Aide à l'Analyse et à la Traduction de Textes Latins"

Sophie JACOB

Promoteur : Cl. Cherton

Mémoire présenté en vue de l'obtention du titre de Licenciée et Maître en Informatique

Je tiens à remercier Monsieur le Professeur Cl. Cherton pour la direction de ce mémoire et ses judicieux conseils tout au long de cette année.

Je remercie également Messieurs M. Absil et J. Denooz pour leur disponibilité et leurs suggestions pertinentes lors de la réalisation de ce travail.

RESUME

Ce mémoire propose une nouvelle technique à utiliser en E.A.O. pour permettre à l'enseignant d'élaborer, au moyen d'un ordinateur, des leçons destinées à être soumises, ensuite, aux élèves, via l'ordinateur.
Une particularité de ce système est qu'il est doté de certaines connaissances spécifiques à la matière enseignée. Toutefois, ce genre de système n'est pas aussi "intelligent" que celui envisagé en I.C.A.L. car il ne contient pas de modèle de l'apprenant.
En utilisant ce procédé, le temps demandé au professeur pour sa préparation est réduit par rapport aux langages auteur actuels. En effet, grâce à son modèle de connaissances, le système génère seul certaines informations composant la leçon. L'intervention de l'enseignant est alors limitée à une description synthétique de la leçon. Seules quelques données supplémentaires inconnues du système doivent être introduites, de manière à ce que ce dernier puisse gérer correctement la phase d'exercices destinée aux élèves. Il est important de remarquer que ces informations sont d'un niveau plus élevé que celles introduites avec des langages auteur, exigeant la description complète de tous les écrans apparaissant à l'élève.

Notre objectif, dans ce travail, était de montrer l'efficacité de ce genre de système. Nous avons, alors, élaboré un outil de démonstration concrétisant nos idées et gérant des exercices d'analyse et de traduction de textes latins. Notre produit n'est qu'une première ébauche, mais laisse déjà préjuger de l'intérêt de ce système. Il nous est permis d'espérer que, pour un travail de l'enseignant moins fastidieux, on puisse aboutir à une leçon d'une qualité meilleure que celle obtenue jusqu'à présent.

SUMMARY

In this thesis, we propose a new method aimed to be used in C.A.L. The use of this method allows a teacher to prepare, with a computer, lessons assigned to be layed students with such a computer.
A particularity of our system is its acquaintance of some specific informations about a subject to teach. However, this kind of system is not so "intelligent" than the one considered in I.C.A.L. The difference between these two processes is the 'student model' : an I.C.A.L. system contains such a model but our system not.
In using our process, the teacher spends less time on his preparation than in using current 'author langagues'. Indeed, in taking advantage of its knowledge model, the system can, itself, generate some components of the lesson. So, the intervention of the teacher is reduced to a synthetic description of the lesson. Only some extra data must be introduced, in order to obtain, from the system, a correct management of the exercices part, assigned to the students. It's important to note that these informations are on a higher level than the ones added by the teacher, using 'author langagues'. Indeed, with these current 'author langagues', all the screens, which are students showed, must be completely described by the teacher.

Our aim, in this work, was to show the efficiency of such a new system. So, we have build up a demonstration product which puts our ideas in concrete form and manages latin texts analysis and translation exercices. Our product is only a first stage in the development of our method, but the interest of the system is yet foreseeable. We are allowed to hope that, for a shorter work asked to the teacher, the quality of the resulted lesson can be beter than the one achieved up to now.

TABLE DES MATIERES

DIDACTICIEL D'AIDE A L'ANALYSE ET A LA TRADUCTION DE TEXTES LATINS

INTRODUCTION

Face au phénomène de société que représente l'introduction croissante des ordinateurs dans des domaines aussi variés que nombreux, le monde de l'enseignement ne reste pas indifférent.
En effet, nous assistons à l'apparition de l'informatique dans les écoles en tant que matière à enseigner, de manière à sensibiliser et former les étudiants à ces techniques modernes.
Remarquons que l'ordinateur se met également au service de l'enseignement et représente un nouvel outil mis à la disposition des pédagogues.

Mais l'ordinateur n'est pas la panacée pédagogique.
Nombreux sont les atouts de l'ordinateur pouvant être exploités d'une manière avantageuse par l'enseignant. Le graphisme, l'animation, la sonorité, la stimulation de la créativité et de l'initiative chez l'élève, telles sont certaines des caractéristiques de l'ordinateur pouvant être intéressantes d'un point de vue pédagogique.
D'un autre côté, il nous faut répondre aux nombreux problèmes soulevés par l'utilisation d'un ordinateur dans l'enseignement. Citons, entre autres, le coût engendré par cette rénovation technique ou la réaction parfois négative des enseignants face à ces machines. Il faut également faire face au risque de vouloir utiliser coûte que coûte ce nouvel outil qu'est l'ordinateur, sous prétexte de le rentabiliser ou de vouloir "être à la page".

Elaborer un didacticiel utile et l'exploiter de manière efficace et satisfaisante pour les étudiants, ne sont pas des tâches évidentes, ni aisées pour les informaticiens et les enseignants.
Cependant, ce sont des tâches valant la peine d'être menées à bien, étant donné les avantages pouvant en ressortir. C'est pourquoi l'E.A.O. est un domaine de recherche informatique à ne pas négliger !

La technique actuellement utilisée dans des didacticiels offrant au professeur la possibilité de préparer lui-même sur ordinateur des leçons destinées à être soumises à ses élèves via cette machine, demande à l'enseignant un travail fastidieux. Nous verrons dans ce mémoire, au cours des chapitres 1 et 2, les caractéristiques, les avantages et les inconvénients de cette technique.
Nous décrirons également, durant ces chapitres, un autre procédé faisant appel à l'intelligence artificielle et exploitant un modèle de la matière à enseigner et un modèle de l'apprenant contenus dans le système. Le champ d'action de ce procédé se limite actuellement à quelques grands domaines privilégiés (ex. médecine). Mais l'utilisation plus fréquente de cette technique dans l'enseignement n'est pas à rejeter pour le futur; c'est pourquoi nous l'examinerons.
Après avoir analysé ces deux techniques, nous ne pourrons en déduire que le fait que chacune d'elles a ses avantages, ses inconvénients et ses domaines d'action. Il n'existe donc pas UN procédé optimal à utiliser dans l'E.A.O. !

Notre objectif, dans ce mémoire, n'est pas de trouver LA solution, mais de proposer une nouvelle méthode aboutissant à des didacticiels permettant aux enseignants de préparer des leçons au moyen d'ordinateurs.
Cette technique demanderait à l'enseignant un travail différent et plus rapide que celui auquel il doit faire face actuellement pour ce genre de tâche. Et, comme nous l'expliquerons aussi au chapitre 3, ce procédé serait également différent de celui exploitant ses deux modèles de connaissances, en faisant appel à l'I.A. En effet, nous verrons que la technique que nous proposons ne considère aucun modèle de l'apprenant dans le système.

Dans le but de prouver l'efficacité d'un tel système, nous avons utilisé cette technique dans l'élaboration d'un didacticiel de démonstration. Celui-ci est spécifique à la langue enseignée qu'est le latin, et plus particulièrement aux leçons et exercices d'analyse et de traduction de textes latins.
Dans le chapitre 4, nous définirons les objectifs généraux d'un tel didacticiel. Le chapitre 5 détaillera ensuite le scénario de l'exploitation de ce système, compte tenu de certaines hypothèses

réduisant le projet de manière à aboutir à un prototype exécutable respectant nos objectifs de démonstration.

Les chapitres 6 et 7, quant à eux, expliciteront, respectivement, le schéma entités-associations conforme à cette application et les données et procédures utilisées lors de l'implémentation.

Les conclusions ressortant de cette recherche et les développements futurs à apporter au produit actuel de manière à l'améliorer et le rendre opérationnel seront finalement définis.

CHAPITRE 1 : UTILISATION DE L'ORDINATEUR COMME OUTIL D'AIDE A L'ENSEIGNEMENT

L'ordinateur est une machine, un outil de travail pluridisciplinaire. Le nombre et la variété des domaines dans lesquels celui-ci peut être utilisé en sont la preuve. Citons, à titre d'exemples, le traitement de textes, l'aide aux handicapés ou l'utilisation de l'ordinateur en tant qu'outil de gestion.

Depuis les années 60, nous assistons à l'introduction de l'informatique dans l'enseignement, en tant qu'outil d'aide à l'enseignement. L'ordinateur devient donc un outil supplémentaire dans l'activité de l'enseignant. Mais il ne donne de bons résultats que s'il est utilisé par quelqu'un de compétent, au bon moment et de la bonne manière.
L'ordinateur en tant qu'outil d'aide à l'enseignement peut être utilisé selon deux grands axes :
il peut soit renforcer les moyens offerts au professeur pour donner ses cours et travaux pratiques, soit être mis à la disposition de l'élève comme système de dialogue individualisé lui permettant d'acquérir et vérifier ses connaissances.
Donc, l'ordinateur peut avoir divers usages pédagogiques.
Premièrement, il peut être un outil de démonstration, de transmission de connaissances, vis-à-vis duquel l'élève ne joue aucun rôle actif, recevant uniquement l'information donnée par la machine. Deuxièmement, l'ordinateur peut aussi être utilisé par l'élève d'une manière conversationnelle comme outil d'exercices, de simulation, ou de "mini-laboratoire" grâce auquel il peut apprendre ou s'exercer.

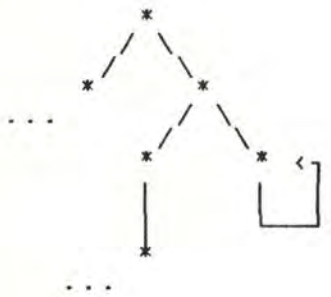
Parmi les autres emplois de l'ordinateur dans l'enseignement, il en existe un au moyen duquel il est possible d'élaborer des systèmes dans lesquels deux modes d'utilisation sont détectés : le "mode enseignant" et le "mode élève". Cela signifie que, d'abord lors d'une première phase, l'enseignant emploie l'ordinateur pour préparer un cours, que ce soit une leçon, des exercices ou même la combinaison des deux. Puis, lorsque cette préparation est terminée, ces cours peuvent être soumis aux élèves par l'intermédiaire d'un dialogue avec un ordinateur.

Remarquons avant de poursuivre, qu'en général, durant cette deuxième phase, la présence d'un professeur n'est pas nécessaire. Celle-ci est, cependant, souvent souhaitable pour aider ou guider l'élève dans son travail. L'intérêt de cette présence est justifié par le fait que l'ordinateur ne remplacera jamais entièrement le pédagogue dans sa fonction.
Divers éléments importants interviennent dans le rôle de l'enseignant durant cette deuxième étape : le moment de son intervention (ex. sur demande ou pas), la nature de ses interventions (ex. explications, guide, ...) et la répartition de son temps disponible entre tous les élèves.

Examinons à présent deux techniques pouvant permettre à l'enseignant d'élaborer ses cours au moyen de l'ordinateur. La première est utilisée de nos jours, tandis que l'autre est plutôt un espoir futur quant à son utilisation fréquente dans le domaine de l'enseignement.

A- La technique utilisée actuellement en E.A.O. :

Cette technique est caractérisée par le fait qu'un ensemble de leçons et d'exercices est schématiquement représenté par un GRAPHE. Les NOEUDS composant ce graphe, représentent chacun un écran contenant un texte, un graphique, une question, ... et sont reliés entre eux par des ARCS. Un ARC entre deux NOEUDS N1 et N2 signifie que l'écran équivalent à N2 s'affiche après celui correspondant à N1.



avec * = écran
/ = transition

Dans le graphe, chaque noeud Ni est à l'origine d'un ensemble de noeuds [Nk], appartenant aussi à ce graphe. Cela signifie qu'un écran correspondant à un noeud de cet ensemble peut s'afficher après l'apparition de l'écran associé au noeud "père" Ni.

Cet ensemble peut être

soit vide : cela signifie que le noeud Ni est terminal dans le GRAPHE. L'écran y correspondant termine un cours donné par l'ordinateur.

soit un singleton [N2] : cela signifie que l'écran équivalent au noeud N2 s'affiche après celui correspondant à Ni, et ce, dans tous les cas, quelle que soit l'intervention de l'élève.

soit un ensemble comprenant au moins deux éléments
= [N2, N3, ..., Nn] avec n >= 3 :
cela signifie qu'un écran correspondant à l'un de ces noeuds s'affiche après celui associé à Ni. Généralement, l'écran associé à Ni contient une question, et c'est suivant la réponse donnée par l'élève que s'affiche tel ou tel écran associé à l'un des noeuds N2, N3, ... , Nn.

Dans un tel système, durant la phase de préparation de leçons et d'exercices, le rôle de l'enseignant est long, même parfois très long, car il doit décrire explicitement et entièrement le graphe associé au cours. L'enseignant doit, en effet, définir précisement tous les noeuds (c'est-à-dire tous les écrans apparaîssant à l'élève), et tous les arcs (c'est-à-dire toutes les transitions entre ces écrans). Cela est dû au fait que le système ne contient aucune connaissance spécifique à quelque matière que ce soit. L'enseignant doit donc introduire lui-même toutes les données composant une leçon particulière, puisque le système ne peut en déduire aucune automatiquement, étant donné son absence d'informations.

Lorsque la phase de préparation est terminée, l'élève peut alors "suivre" un de ces cours préparés par le professeur. Il s'agit d'un affichage d'écrans consécutifs, dépendant des réponses données par cet élève. Une leçon donnée à un élève est donc un CHEMIN dans le graphe élaboré par l'enseignant.

Mentionnons qu'un outil a été spécialement conçu pour l'enseignant de manière à faciliter la construction de tels cours. Il s'agit des langages auteur. Un LANGAGE AUTEUR est un langage de programmation qui simplifie l'écriture des leçons et des exercices, en offrant à l'enseignant des "facilités", telles que des instructions permettant, par exemple, la gestion d'un

écran ou le traitement des réponses par mots-clés.

Remarquons également que certains didacticiels d'E.A.O. permettent la gestion de l'historique du travail de l'élève. Bien que ce traitement soit, en général, assez rudimentaire, il est toutefois intéressant à exploiter d'un point de vue pédagogique. La possibilité pour l'enseignant d'interroger le système à propos du déroulement des leçons et des exercices donnés aux élèves, offre à celui-ci des informations lui permettant d'adapter ses cours.
Un autre intérêt de l'historique du travail de l'élève est de pouvoir en faire dépendre les enchainements d'écrans proposés à l'élève. En général, seule la réponse donnée par l'élève à la question affichée influence l'apparition de l'écran suivant. Cependant, il serait possible de faire dépendre l'affichage d'un écran d'un historique à plus long terme, donc de plus d'informations qu'uniquement la dernière réponse. Pour cela, il serait envisageable d'exploiter les possibilités offertes par l'ordinateur, telles que le traitement de chaînes de caractères ou de calculs. Cependant, ces traitements n'ont qu'une portée générale et rien de spécifique à la branche enseignée ne peut être géré par le système. Il n'est donc pas souvent possible, par ce procédé, d'interprêter toutes les réponses déjà données par l'élève en termes de connaissances acquises par celui-ci.

B- La technique utilisée en I.C.A.L. (Intelligent Computer Aided Learning)

Ce procédé diffère du premier par le fait qu'un tel système, faisant appel à l'intelligence artificielle, contient un modèle de connaissances de la matière enseignée et un modèle de l'apprenant. De plus, le système sait exploiter ces deux modèles. Dès lors, le travail de l'enseignant en est simplifié : il n'a plus à introduire les informations déjà connues du système.
Actuellement, l'exploitation de cette technique n'en est qu'à ses débuts. Seuls certains domaines d'application, telle la médecine, composent son champ d'action. Mais, la propagation de ce procédé dans l'enseignement trouverait son intérêt en fournissant aux enseignants des outils spécifiques en rapport avec le modèle de connaissances contenu dans le système.

CHAPITRE 2 : "CRITIQUES" DE CES PROCEDES INFORMATIQUES D'ELABORATION DE LECONS

A- La technique utilisée actuellement en E.A.O.

Ce procédé présente l'avantage d'offrir à l'enseignant la possibilité de décrire explicitement toutes les informations composant ses cours (écrans et transitions) exactement comme il désire qu'elles apparaissent à l'élève.
Mais le revers de la médaille survient rapidemment. En effet, les cours donnés par l'ordinateur à l'élève ne sont composés que des informations introduites par l'enseignant, aucune autre donnée n'y apparaît en plus de celles du professeur. Cela se manifeste aussi bien au niveau du contenu des leçons, qu'à propos de "la forme" de ces leçons, c'est-à-dire de la disposition des données sur l'écran, des questions posées, ... Cela est dû au fait que le système n'est pas capable de générer quoi que ce soit dans une leçon ou un exercice, puisqu'il ne possède aucun modèle de connaissances relatives à la matière à enseigner.
Donc, non seulement le temps demandé à l'enseignant pour préparer ses cours sur ordinateur est énorme, car il doit décrire chaque élément appartenant au graphe représentant la leçon, mais, en plus, seuls les cas prévus par l'enseignant sont pris en compte dans cette leçon. En général, la contrainte temporelle de préparation rend les cours élaborés par ce procédé relativement incomplets.

Cette technique est, dès lors, très attrayante en théorie, car elle permet à l'enseignant de "faire tout ce qu'il veut". Mais, en pratique, elle devient inefficace, car le temps demandé à l'enseignant pour préparer un cours est énorme pour un résultat qui n'est pas d'une qualité extraordinaire.

B- La technique utilisée en I.C.A.L.

L'usage dans l'enseignement de la technique contenant les modèles de connaissances de la matière à enseigner et de l'apprenant, tout en faisant appel à l'intelligence artificielle, permettrait de pallier le mauvais rendement du procédé actuellement utilisé en E.A.O.
Il est certain que la présence dans le système du modèle de connaissances de la matière à enseigner rend cet outil plus spécifique que celui utilisé actuellement en E.A.O., qui lui, a

un champ d'utilisation plus général. Mais cette spécificité ne joue qu'en son avantage, car le fait que le système soit capable d'exploiter ses connaissances à propos de la matière enseignée réduit le temps et le travail de préparation de l'enseignant. Celui-ci ne doit, en effet, plus introduire les données connues du système. De plus, grâce à ce modèle, un tel système peut détecter avec précision une réponse erronée de l'élève.
Le modèle de l'apprenant, quant à lui, contient des informations diverses caractérisant les comportements, la psychologie et les profils typiques des élèves : c'est un "modèle d'apprentissage". Grâce à ces données, le système peut, non pas expliquer les erreurs de l'élève, mais en déterminer la nature : il peut s'agir d'erreurs d'inattention, de compréhension ou d'un tout autre genre. A partir de ces renseignements, le système peut, alors, automatiquement adapter, d'une manière efficace, ses cours à tel ou tel élève selon le comportement de celui-ci.

Du fait de la spécificité de chaque système, il est certain que plusieurs systèmes sont nécessaires pour traiter plusieurs matières. Il ne sera donc possible d'élaborer un cours au moyen de cette technique que s'il existe un système contenant un modèle de la matière à enseigner en question. Cette exigence de modélisation est, pour l'instant, une limite quant à la diversité des matières pouvant être enseignées par cette technique. En effet, certaines branches ne sont actuellement pas modélisées (cela peut être dû soit à nos limites courantes en informatique, soit au fait que nous pensons, pour l'instant, que ce domaine n'est pas formalisable). Mais, rappelons que cette technique n'est qu'au début de son développement. Cette limite actuelle ne peut donc être qu'allégée par la suite. La modélisation de nouvelles matières permettra la création de nouveaux didacticiels utilisant la technique d'I.C.A.L.

Ce procédé est, par conséquent, fort attrayant d'un point de vue pratique, car il est plus "productif" : à temps de préparation égal avec les langagues auteur, la qualité du didacticiel obtenu est bien meilleure grâce à cette technique. Sa difficulté, cependant, consiste en l'élaboration d'un modèle de connaissances de la matière à enseigner et d'un modèle de l'élève, ainsi qu'en leur exploitation.

CHAPITRE 3 : AUTRE TECHNIQUE D'ELABORATION DE LECONS AU MOYEN DE L'ORDINATEUR : DESCRIPTION ET OBJECTIFS GENERAUX

L'objectif à atteindre est de résoudre les défauts des systèmes actuels d'E.A.O. utilisant les langagues auteur, c'est-à-dire réduire le temps de préparation demandé à l'enseignant et augmenter la qualité des cours élaborés. Voilà pourquoi nous proposons, à présent, une autre technique permettant de créer des leçons et des exercices via un ordinateur.
L'originalité de ce système par rapport aux deux autres techniques décrites précédemment est qu'il peut être considéré comme un intermédiaire entre ces deux possibilités.
En effet, un tel système différera de celui actuellement utilisé en E.A.O., car il contiendra un modèle de la matière enseignée. De plus, le système saura exploiter ces connaissances, ce qui simplifiera le travail de préparation de l'enseignant et donc en réduira le temps requis. Les interventions du professeur ne seront plus aussi fastidieuses qu'actuellement, avec les langagues auteur. Elles se limiteront à l'ajout de certaines informations supplémentaires par rapport au modèle contenu dans le système et inconnues de celui-ci.
D'un autre côté, un tel système sera différent de celui d'I.C.A.L. par le fait qu'il ne considérera en aucune façon un modèle de l'élève.

Remarquons que, tout comme il serait possible à l'élève de deviner si le cours que l'ordinateur lui soumet a été préparé par la technique actuelle utilisée en E.A.O. ou par celle d'I.C.A.L., il pourrait aussi détecter si ce cours a été élaboré par cet autre système proposé. En effet, considérons comme point de comparaison les messages s'affichant en réponse à une erreur de l'élève :

- un cours résultant d'un langague auteur actuel d'E.A.O. ne contient que les informations introduites par l'enseignant. Or, comme ce travail de préparation demande énormément de temps au professeur, celui-ci, bien souvent, ne détaille pas beaucoup les messages apparaissant à l'écran après une erreur commise par l'élève. Ces messages sont donc souvent assez froids et stéréotypés.

- quant aux cours élaborés par un système d'I.C.A.L., ils permettent un traitement des erreurs plus précis grâce au modèle de connaissances de la matière enseignée. Le système peut, généralement, détecter une erreur automatiquement avec précision et afficher un message adéquat. De plus, ce message peut être adapté à chaque élève grâce au modèle de l'apprenant contenu dans ce système. Dans le cas où les connaissances du système ne lui permettent pas de gérer correctement les erreurs de l'élève, c'est à l'enseignant d'intervenir pour introduire les messages adéquats.

- en ce qui concerne cet autre système que nous proposons, il pourra aussi générer automatiquement des messages précis relatifs aux erreurs commises, cela, grâce à son modèle de connaissances de la matière à enseigner. Mais, le système ne contient pas de modèle de l'apprenant, les messages tenant compte du comportement de l'élève ou ceux non générés correctement ou non assez détaillés par le système sont alors introduits par le professeur.

Vue par l'enseignant, la différence entre l'utilisation de ce système et celle des autres techniques d'E.A.O. et d'I.C.A.L. sera encore plus remarquable.
En effet, par rapport au système actuel d'E.A.O., on devrait noter une baisse du temps de préparation et une diminution des interventions de l'enseignant pour un résultat au moins équivalent à celui obtenu actuellement. Utilisant cette technique, le professeur ne devra plus décrire explicitement tout le graphe, le réseau d'écrans représentant un ensemble de leçons et d'exercices. Les interventions de l'enseignant se situeront à un autre niveau de détails. Ce ne sera plus en termes d'écrans de dialogue avec l'élève qu'il raisonnera et agira, mais à un niveau plus élevé, en termes d'informations et de notions utilisées dans ce dialogue. Insistons sur le fait qu'une grande partie des informations composant le dialogue avec l'élève sera générée automatiquement par le système à partir de son modèle de connaissances de la matière à enseigner. L'enseignant n'aura donc plus à ajouter qu'un nombre réduit de données supplémentaires non fournies par le système. A partir de toutes ces informations, le système générera le dialogue précis qu'il aura avec un élève. Ceci représente une différence importante avec les langages auteur où l'ENSEIGNANT élabore TOUT le graphe

correspondant à un ensemble de leçons et d'exercices. Dans notre cas, le SYSTEME n'établira qu'un CHEMIN ou parfois un ensemble de chemins de ce graphe correspondant uniquement au cours suivi par un élève. De plus, cette élaboration se réalisera lors de la phase de travail avec l'élève, et non durant la phase de préparation comme cela se passe actuellement.

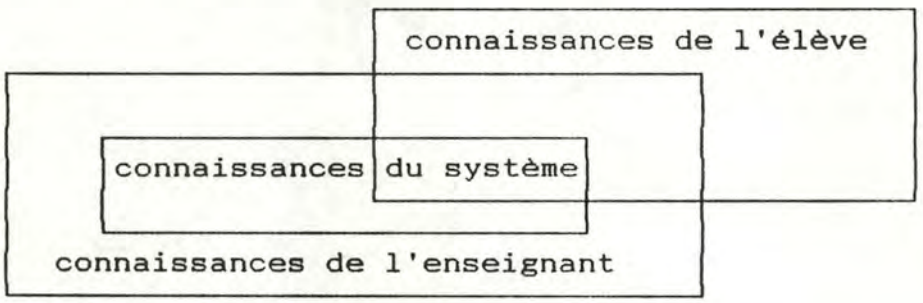
Un intérêt de ce procédé est donc qu'il simplifie le travail de l'enseignant, mais l'inconvénient y associé est qu'il empêche le professeur d'agir à un niveau atomique, sur un détail précis d'un certain écran.

Un autre avantage de ce procédé est le fait que l'enseignant peut préparer, s'il le désire, un cours spécifique pour chaque élève. Cela est possible car le professeur n'a pas à introduire toutes les informations utilisées dans chacun de ces cours, mais uniquement à ajouter certaines données nécessaires et inconnues du système. Cela ne serait pas envisageable en pratique en utilisant la technique actuelle d'E.A.O. En effet, étant donné que le temps de préparation est déjà important pour un cours général, il n'est pas réaliste de demander à l'enseignant d'élaborer de cette manière un cours spécifique pour chaque élève.

On pourrait imaginer que la base de connaissances de la matière à enseigner contenue dans notre système lui permette de générer seul automatiquement toutes les informations nécessaires à l'élaboration d'un ensemble de cours. Cela serait certainement rare. Mais même si c'était le cas, en général, l'enseignant désirera intervenir pour compléter le travail du système et en améliorer encore la qualité pédagogique ou y apporter des renseignements pour l'adapter à un élève particulier.
Quelle que soit la raison pour laquelle l'enseignant interviendra, il mentionnera des informations supplémentaires inconnues du système. C'est-à-dire que face aux résultats proposés par le système, à partir de ses connaissances, le professeur pourra décider d'en évincer certains qui sont incorrects pour une raison inconnue du système. L'enseignant aura également la possibilité d'ajouter des commentaires explicatifs associés à ses interventions et destinés aux élèves.

Une question pourrait se poser concernant ce procédé : "Quid du traitement des réponses fausses d'un élève ?"
Nous savons, que dans un cours établi par une technique actuelle d'E.A.O., une mauvaise réponse fournie par un élève peut être expliquée par un enchainement spécifique de questions-réponses, pour autant que l'enseignant ait prévu cette réponse incorrecte.
Notre nouveau procédé le permet-il aussi ?
Notre système sait expliquer une bonne réponse en se basant sur ses connaissances, mais il ne peut pas expliquer une mauvaise réponse, car, pour cela, il devrait contenir un modèle des erreurs des élèves. Etant donné que ce n'est pas le cas, la réaction du système face à une mauvaise réponse d'un élève sera, généralement, l'apparition d'un message généré automatiquement et mentionnant simplement l'erreur commise.
Cependant, nous savons que l'enseignant intervient lors de la phase de préparation pour ajouter des informations inconnues du système. Nous pourrions alors imaginer que si l'enseignant veut gérer une mauvaise réponse, d'une manière plus détaillée que celle du système, deux possibilités seraient à sa disposition. Premièrement, il pourrait introduire lui-même un message explicatif plus précis que celui du système relatif à l'erreur commise. Ce traitement de l'erreur se limiterait, alors, à l'affichage d'un commentaire et non à un enchainement découlant de la faute. Deuxièmement, l'enseignant pourrait ajouter lui-même toutes les informations non générées par le système et nécessaires à un enchainement de questions-réponses, ayant pour but d'expliquer l'erreur d'une manière encore plus détaillée qu'un simple message. Ce deuxième traitement serait davantage basé sur l'idée de faire découvrir à l'élève sa faute : celui-ci devrait s'en rendre compte lors de cet enchainement.
Cependant, une erreur d'un élève peut être due soit à un manque d'informations, soit à une information erronée. Dès lors, donner à l'enseignant la possibilité de gérer des erreurs du deuxième type, en ajoutant des informations pour générer un enchainement questions-réponses, signifierait que le professeur introduirait dans le système les fautes de l'élève afin de les traiter. Or, cela ne serait pas cohérent avec le modèle de connaissances de la matière contenu dans le système. Cela représenterait, en fait, un modèle des erreurs de l'élève, et le traitement de ce modèle appartient à une toute autre approche que celle que nous considérons dans cette technique.

Abandonnons donc cette idée de donner à l'enseignant la possibilité d'ajouter de "fausses" informations dans le système, dans le but de gérer des réponses d'un élève, erronées pour cause de renseignements incorrects. L'enseignant ne pourra, alors, qu'accepter ou refuser les solutions proposées par le système, il ne pourra pas en ajouter d'autres.
Remarquons, quand-même, que les erreurs dues à un manque d'informations pourront être normalement traitées par le système. En effet, celui-ci, à partir de ses connaissances correctes mais incomplètes, pourra génèrer, seul, sans le savoir, des informations correspondant à un enchainement "faux et explicatif" découlant d'une erreur provoquée par un manque d'informations. En fait, le système commettra la même faute que l'élève qui donnera cette réponse erronée !
Le schéma suivant illustrant les connaissances du système, de l'enseignant et de l'élève nous permet de mieux comprendre cette gestion des erreurs de l'élève.



Baptisons le nouveau procédé proposé et appelons-le désormais "technique semi-automatique". Ce nom est justifié par le fait le système génère une partie des informations composant la leçon, en se basant sur ses connaissances, et qu'ensuite, l'enseignant complète ce travail automatique.

CHAPITRE 4 : APPLICATION DE LA TECHNIQUE "SEMI-AUTOMATIQUE" : "DIDACTICIEL D'AIDE A L'ANALYSE ET A LA TRADUCTION DE TEXTES LATINS" : DEFINITION ET OBJECTIFS GENERAUX DU PROJET CADRE

Nous allons essayer d'élaborer un "didacticiel d'aide à l'analyse et à la traduction de textes latins" selon la technique "semi-automatique". Ce choix de l'application est justifié par le fait que le latin est une langue possédant une grammaire assez développée (conjugaisons, déclinaisons,...). Cette connaissance, partiellement introduite dans le système, permettra à celui-ci d'effectuer certaines tâches d'une manière automatique. Ces résultats seront ensuite complétés par l'enseignant. Ceci répond donc bien aux spécificités de la technique "semi-automatique".
Le mémoire, réalisé en 1986 par J. Frippiat, s'attachait à l'analyse d'un logiciel d'E.A.O. "guide à la traduction de textes latins ou grecs". Nous adapterons ses idées à notre démarche et viserons, à présent, à la réalisation de ce logiciel en nous limitant à la langue latine.

Ce didacticiel permettra de "gérer" des exercices d'analyse et de traduction de textes latins en deux phases. D'abord, l'enseignant préparera ces exercices, tout en étant en grande partie aidé par les résultats générés automatiquement par le système, à partir de son modèle de connaissances du latin. Puis, ces exercices seront proposés aux élèves.
Le système traitera ces deux phases d'une manière analogue, et ces traitements correspondront à la vue qu'ont l'enseignant et l'élève concernant un exercice d'analyse et de traduction d'un texte latin. Cela est important car, pour qu'un didacticiel soit bien accepté, utilisé et efficace, il doit représenter au mieux le raisonnement naturel de l'utilisateur.

L'objet sur lequel portera un exercice d'analyse et de traduction sera un TEXTE LATIN. Un texte latin est une suite de PHRASES LATINES, elles-mêmes représentant une suite de MOTS LATINS.
Nous supposerons, dans le cadre de notre travail, qu'il n'existe aucune interaction entre ces phrases. Nous envisagerons donc toujours une seule phrase à la fois.

Nous émettrons également les hypothèses suivantes :
- la structure d'une phrase latine est relativement simple
- le vocabulaire employé est assez restreint
- aucune faute d'orthographe n'est commise
  (la gestion des fautes d'orthographe, dans ce qu'introduit l'enseignant ou l'élève, serait intéressante à traiter afin d'établir une correction automatique de ces erreurs. Espérons que cela fasse l'objet d'une extension future du didacticiel.)

A présent, après avoir défini l'objet sur lequel portera un exercice, décrivons le traitement réalisé sur ces textes latins : leur analyse.
L'ANALYSE D'UN TEXTE LATIN équivaut à l'analyse de chaque phrase de ce texte. Or, étant donné que nous avons précisé que chaque phrase est considérée comme sans interaction avec les autres, nous pouvons également dire que les analyses de ces phrases sont indépendantes l'une de l'autre. Dès lors, poursuivons notre raisonnement en considérant le cas général de l'ANALYSE D'UNE PHRASE LATINE.
L'analyse complète d'une phrase latine se base sur quatre points de vue : la syntaxe, la morphologie, la sémantique et la traduction. Nous évincerons, cependant, dans notre travail, l'aspect sémantique du modèle de connaissances du système, car la gestion de cet aspect est un problème fort complexe. En effet, les logiciels intégrant un peu de sémantique portent sur des contextes très restreints, cela est dû à la difficulté de modélisation de cet aspect. Nous avons décidé de n'intégrer aucune connaissance sémantique dans notre système. Ce sera du ressort de l'enseignant de fournir, si nécessaire, les informations supplémentaires dépendant de cette sémantique.

Dès lors, la base de connaissances du système se composera d'informations concernant la syntaxe d'une phrase latine, la morphologie des mots latins et leurs traductions. A partir de ce modèle, le système pourra générer une grande partie des informations et messages qui composeront le dialogue avec l'élève. Le rôle de l'enseignant sera limité à certains compléments à apporter aux résultats du système, ces compléments dépendront notamment de la sémantique.

Schématiquement, nous pouvons caractériser la phase de préparation d'un ensemble d'exercices d'analyse et de traduction de textes latins, de la manière suivante :

- des textes latins seront édités par l'enseignant, via un quelconque traitement de textes, tout en respectant, éventuellement, certaines conditions de présentation et utilisation (ex. pas de caractères de commandes, uniquement une suite de caractères "lisibles")
- le système les analysera automatiquement "off-line". C'est-à-dire, qu'à partir de ses connaissances, il générera l'analyse de chaque phrase de chaque texte : leurs analyses syntaxiques, les analyses morphologiques de chacun de leurs mots, ainsi que leurs traductions
- l'enseignant aura, ensuite, la possibilité de compléter les éléments fournis par le système : il pourra confirmer ou infirmer certaines étapes de l'analyse proposées par le système et il pourra aussi confirmer, infirmer ou ajouter des messages destinés à l'élève
- puis, les interventions de l'enseignant seront "vérifiées" par le système. Et, selon que certaines incohérences seront détectées ou pas, l'enseignant devra à nouveau intervenir pour corriger ces incohérences. Ensuite, lorsque la phase de préparation d'un exercice d'analyse et de traduction d'un texte latin sera terminée, cet exercice pourra être soumis aux élèves.

CHAPITRE 5 : "DIDACTICIEL D'AIDE A L'ANALYSE ET A LA TRADUCTION DE TEXTES LATINS" : REDUCTION DU PROJET CADRE ET DETAILS DU SCENARIO

Etant donné que ce "didacticiel d'aide à l'analyse et à la traduction de textes latins" sera original par rapport aux autres logiciels déjà existants en E.A.O., puisqu'il utilisera cette technique "semi-automatique", l'objectif principal de ce mémoire est d'analyser la faisabilité de ce genre de logiciel.
Pour cela, nous testerons certains critères : la facilité de préparation pour l'enseignant, le temps de préparation, la qualité des leçons établies, ...

Mais avant de tester cette faisabilité, il faut disposer du produit sur lequel s'effectueront ces tests. Le but principal de notre travail ne consistant pas à l'élaboration du didacticiel opérationnel, nous nous limiterons à la réalisation d'un outil de démonstration correspondant à l'application définie au chapitre précédent.
Nous savons que celui-ci devra essayer de générer automatiquement les analyses syntaxiques des phrases latines, les analyses morphologiques des mots latins, leurs traductions et les messages associés à ces étapes. Nous nous sommes alors renseignés sur l'existence d'analyseurs morphologiques et syntaxiques en langue latine. Cela, non pas parce que nous ne voulons ou ne pouvons pas les réaliser nous-mêmes, mais, dans l'espoir de les intégrer dans notre logiciel de démonstration, afin de gagner du temps dans l'élaboration de celui-ci. Cette recherche est justifiée puisque notre objectif premier n'est pas vraiment l'élaboration de ces analyseurs, mais l'indication de l'efficacité du système utilisant, entre autres, ces analyseurs.

Concernant l'**analyseur syntaxique** d'une phrase latine, il n'en existe pas qui réalise cette analyse d'une manière non-interactive. En effet, une phrase latine ne possède pas de structure syntaxique fixe, la place des mots dans une telle phrase n'est pas davantage figée ni fonctionnelle et la forme des mots latins n'est pas suffisante non plus pour établir automatiquement l'analyse syntaxique d'une phrase latine. Outre ces faits, un autre élément important compliquant cet automatisme est la SEMANTIQUE. Comme nous l'avons déjà souligné, cet aspect est difficilement traitable par le système. Dès lors, même un

analyseur syntaxique assez complexe ne saura jamais éliminer seul toutes les mauvaises analyses syntaxiques d'une phrase latine. Ces problèmes justifient donc l'intervention de l'enseignant, lors de la phase d'élaboration des analyses syntaxiques.
De cette manière, en considérant que le système contient un analyseur syntaxique relativement "rudimentaire" mais interactif, nous pouvons espérer un résultat de bien meilleure qualité que s'il n'était pas interactif. Car, grâce aux renseignements donnés par l'enseignant, les analyses syntaxiques sémantiquement erronées seront rapidement éliminées.

Quant à l'**analyseur morphologique**, il en existe un travaillant de manière non-interactive. Il a été réalisé par le L.A.S.L.A. (Laboratoire d'Analyse Statistique des Langues Anciennes) de l'Université de Liège. Le professeur J. Denooz tient à ce que Liège reste propriétaire de ce programme, fruit de leur labeur. Cependant, il nous permet d'utiliser les résultats de son logiciel, ce qui nous sera très utile. Cela signifie donc que nous travaillerons, dans le cadre de ce mémoire, sur des textes déjà analysés morphologiquement par le L.A.S.L.A. de l'Université de Liège.

Redéfinissons, à présent, le concept de TEXTE LATIN qui est l'objet d'exercices d'analyse et de traduction :
un TEXTE LATIN est une suite de PHRASES LATINES qui sont, chacune, une suite de MOTS LATINS auxquels sont associées les ANALYSES MORPHOLOGIQUES élaborées par le L.A.S.L.A. de l'Université de Liège. Ces textes ne sont, par conséquent, plus quelconques, comme cela l'était prévu dans le Projet Cadre. Ils ne sont plus introduits par l'enseignant au moyen du clavier. Les textes sur lesquels le système pourra travailler sont uniquement ceux fournis par le L.A.S.L.A. de l'Université de Liège, donc des textes dont chaque mot est analysé morphologiquement.

Le système que nous avons l'intention de mettre en oeuvre n'est, dès lors, plus aussi "intelligent" que celui prévu dans le Projet Cadre. Il ne contient plus d'analyseur morphologique, ni d'analyseur syntaxique automatiques.
Notre tâche dans la réalisation du didacticiel consiste alors à élaborer un analyseur syntaxique rudimentaire et interactif, à exploiter les informations morphologiques qui sont à notre disposition, à gérer un lexique latin-français et surtout à

intégrer les compléments introduits par l'enseignant lors de la préparation de l'analyse.

Schématiquement, le déroulement de la phase de préparation d'exercices d'analyse et de traduction des textes latins dans notre didacticiel de démonstration est le suivant :

- des textes latins, auxquels sont associées les analyses morphologiques de chacun de leurs mots, nous sont fournis par le L.A.S.L.A. de l'Université de Liège

- aucune analyse automatique "off-line" n'est générée par le système à ce niveau, étant donné que l'analyse morphologique a déjà été réalisée par le L.A.S.L.A. de l'Université de Liège et que l'analyse syntaxique n'est pas traitable non-interactivement

- l'enseignant complète alors les propositions d'analyse du système. Il peut confirmer ou infirmer certaines solutions proposées quant aux analyses morphologiques et aux traductions. Il peut aussi confirmer, infirmer ou ajouter des messages associés à chaque étape d'analyse. Et c'est aussi à ce moment-là que l'analyse syntaxique est élaborée interactivement, à partir des propositions du système et des informations supplémentaires de l'enseignant

- les interventions de l'enseignant sont ensuite "vérifiées" par le système, et selon que certaines incohérences ont été détectées ou pas, l'enseignant doit à nouveau intervenir pour les corriger. Finalement, lorsque la phase de préparation d'un exercice d'analyse et de traduction d'un texte latin est terminée, cet exercice peut être soumis aux élèves.

5.1- Elaboration d'un exercice d'analyse et de traduction d'un texte latin

Attardons-nous, à présent, sur la phase de préparation par l'enseignant d'un exercice d'analyse et de traduction d'un texte latin.
Comme nous l'avons déjà dit, l'analyse d'un texte latin est, en fait, l'analyse de chaque phrase de ce texte. Et l'analyse complète d'une phrase consiste en la description de ses analyses syntaxiques, des analyses morphologiques de chaque mot de cette phrase, ainsi que des traductions de chacun d'eux. Nous ne procéderons pas à l'élaboration de chaque partie séparément et indépendamment l'une de l'autre, nous établirons plutôt l'analyse totale en combinant les trois points de vue, ce qui correspond davantage à un raisonnement naturel.

5.1.1- Comment gérer l'analyse complète d'une phrase latine ?

Considérons, à titre d'illustration, une phrase écrite dans une langue quelconque et dont la structure syntaxique est la suivante : "GNsujet V GNcomplément", sachant que le verbe est le pilier de la phrase. Mentionnons tout de suite que ceci n'est pas toujours le cas. En effet, généralement une phrase est caractérisée par un verbe principal, mais ce n'est pas une règle standard. L'élément de base d'une phrase peut aussi être
soit un substantif : ex. Où est la balle ? Dans le jardin .
soit un adjectif : ex. Comment trouves-tu ces fleurs ? Jolies .
soit un numéral : ex. Combien as-tu de bonbons ? Deux .
soit un adjectif pronom : ex. Lequel préfères-tu ? Celui-là .
soit un adverbe : ex. Où ?
soit une préposition : ex. Où mettre les fleurs par rapport à la table ? Dessus .
soit une conjonction .

Ceci étant précisé, voici quelle est la démarche naturelle d'analyse d'une telle phrase :

- à partir de la phrase entière, nous cherchons tout d'abord le verbe de base, pilier de cette phrase
- lorsque celui-ci a été détecté, nous en établissons l'analyse morphologique. Puis, nous pouvons alors considérer la phrase non plus comme une seule suite de mots, mais comme étant composée de

trois parties principales : le verbe, le groupe nominal sujet de ce verbe et le groupe nominal complément de ce verbe
- chaque groupe nominal est ensuite analysé selon un niveau de détails plus raffiné. Nous traitons chaque morceau de phrase en sélectionnant, tout d'abord, le mot formant la base de ce morceau de phrase. Puis, nous analysons morphologiquement ce mot de base, et ensuite, nous y rattachons tous les mots y associés de manière à déterminer syntaxiquement, entièrement le morceau de phrase et sa fonction dans la phrase. Chaque mot sélectionné est, alors, analysé morphologiquement.

Nous détaillons ainsi l'analyse de la phrase jusqu'à avoir traité et, finalement, traduit chaque mot de cette phrase à analyser.

Si nous considérons une phrase latine de même structure que celle précitée, nous pouvons dire que son analyse suit la même démarche que celle que nous venons de décrire et correspond, donc, à une suite d'étapes respectant un ordre de traitement naturel :
sélection du verbe principal, analyse morphologique de ce verbe, sélection de la base du groupe nominal sujet, analyse morphologique de celle-ci, détermination de tous les mots composant le groupe nominal sujet, ...

Nous pouvons généraliser cette illustration et considérer une phrase de structure syntaxique quelconque. Son analyse est caractérisée par des étapes du même genre que celles venant d'être décrites, à savoir, sélection d'un mot correspondant à une certaine fonction, analyse morphologique d'un mot, découpe syntaxique d'un morceau de phrase et finalement traduction de cette phrase analysée.

Remarquons, à présent, que lors d'un exercice d'analyse et de traduction d'une phrase latine, l'élève peut ne pas toujours trouver immédiatement toutes les étapes d'analyse correctes. De plus, certaines étapes d'analyse peuvent être possibles à un certain niveau d'analyse et s'avérer incorrectes uniquement par la suite. La considération d'étapes d'analyse incorrectes nous semble, dès lors, intéressante à traiter auprès de l'élève.

Dans notre travail, l'analyse d'une phrase latine est gérée de la manière suivante :
Les analyses possibles (correctes ou incorrectes) d'une phrase latine sont représentées sous la forme d'un ARBRE D'ANALYSE. Chaque NOEUD de cet arbre contient des informations associées à une étape possible d'analyse. Dans cet ARBRE D'ANALYSE, certaines BRANCHES représentent des analyses incorrectes de la phrase, tandis que d'autres correspondent à celles exactes.

Lors de la phase de préparation, l'enseignant participe à l'élaboration des arbres d'analyse associés aux phrases latines à analyser. Or, nous savons que le modèle de connaissances d'une matière à enseigner contenu dans le système, est très rarement complet. Ceci est particulièrement vrai dans notre cas. Certaines étapes d'analyse proposées par le système sont incorrectes. Cette inexactitude est due à un manque d'informations du système. Il est fort possible, pourtant, que l'élève, pour les mêmes ignorances, aboutisse à cette réponse fausse. L'enseignant peut, alors, avoir deux types de comportements face à chacun de ces noeuds représentant une étape d'analyse incorrecte proposée par le système :

- s'il estime qu'il n'est pas possible de se rendre compte de l'inexactitude de cette étape à ce niveau d'analyse de la phrase, il qualifie alors le noeud correspondant à cette étape d'analyse d'"indicatif". Ce qui signifie que cette étape incorrecte est momentanément acceptée par l'enseignant. L'élève ayant effectué cette étape pourra continuer, s'il le désire, son analyse dans cette voie. L'objectif poursuivi dans ce cas étant qu'il comprenne lui-même son erreur, en réalisant qu'en développant son analyse ainsi commencée, il n'aboutira à rien de valable.
(Remarquons qu'un noeud correspondant à une étape d'analyse correcte sera aussi qualifié d'"indicatif", permettant ainsi à l'élève de poursuivre son analyse dans cette voie.)

- s'il estime, en revanche, que cette étape d'analyse erronée n'est pas admissible, il associe alors au noeud y correspondant la mention "break". Ceci signifie que le chemin aboutissant à ce noeud est sans issue et ne représente pas une analyse correcte de la phrase. Lors d'un exercice d'analyse et de traduction de la phrase en question, l'élève ayant réalisé cette étape d'analyse erronée ne pourra alors pas continuer son analyse en suivant

cette voie, car il aurait dû se rendre compte de son erreur.

Donc, à chaque noeud de l'arbre d'analyse sera associé un "type d'étape" qui aura la valeur "indicatif" ou "break".

Etant donné que chaque NOEUD contient des informations associées à une étape possible d'analyse d'une phrase latine, et que ces étapes sont de différentes natures, selon le niveau de l'analyse, les noeuds y associés sont donc de différents types et contiennent différentes sortes d'informations, selon leur type. Les différentes caractéristiques d'un noeud sont les suivantes :

- type du noeud :
  '1' = choix d'un mot correspondant à une fonction
  (ex. sélection du verbe principal)
  '2' = analyse morphologique d'un mot
  (ex. analyse morphologique du verbe principal)
  '3' = analyse syntaxique
  = association d'un ensemble de mots à un symbole catégoriel
  (ex. détermination de l'ensemble des mots de la phrase représentant le groupe nominal sujet)
  '4' = traductions admises de tous les mots de la phrase selon un certain sens

- commentaire élaboré automatiquement par le système, d'après notamment le type du noeud

- commentaire de l'enseignant associé à ce noeud et à afficher, à la place de celui du système, si l'élève considère les informations composant ce noeud comme réponse, à un moment donné de l'analyse

- mention d'acceptation de l'affichage du commentaire correspondant à ce noeud : soit celui du professeur, soit celui du système si celui de l'enseignant est inexistant

- type de l'étape d'analyse : "indicatif" ou "break"

- référence au noeud père, à l'étape d'analyse précédente

- booléen mentionnant l'appartenance ou la non appartenance de ce noeud à un chemin représentant une analyse correcte de la phrase traitée

- informations composant le noeud selon son type :

si '1' : un mot
et la fonction y associée (V, Sujet, ...)

si '2' : catégorie d'un mot
et caractéristiques y associées
(ex. V, temps, mode, pers, nombre, ... ;
Subst, cas, genre, ... ;
...)

si '3' : symbole catégoriel syntaxique (GN, GV, V, ...)
et ensemble des mots y associé

si '4' : toutes les traductions françaises possibles de chaque mot de la phrase, associées à une confirmation ou une infirmation d'acceptation contextuelle.

Cette structure d'ARBRE D'ANALYSE est remplie, en partie, par le système et, en partie, par l'enseignant. En effet, respectant les caractéristiques de ce nouveau genre de didacticiel envisagé, une grande partie du travail de préparation de l'enseignant, concernant dans notre cas l'analyse de textes latins, est établie automatiquement par le système, à partir de ses connaissances. Ensuite, l'enseignant complète ces résultats.

Avant de poursuivre, comparons l'arbre d'analyse d'une phrase latine que nous venons de définir et la structure du réseau d'écrans utilisée actuellement en E.A.O. :

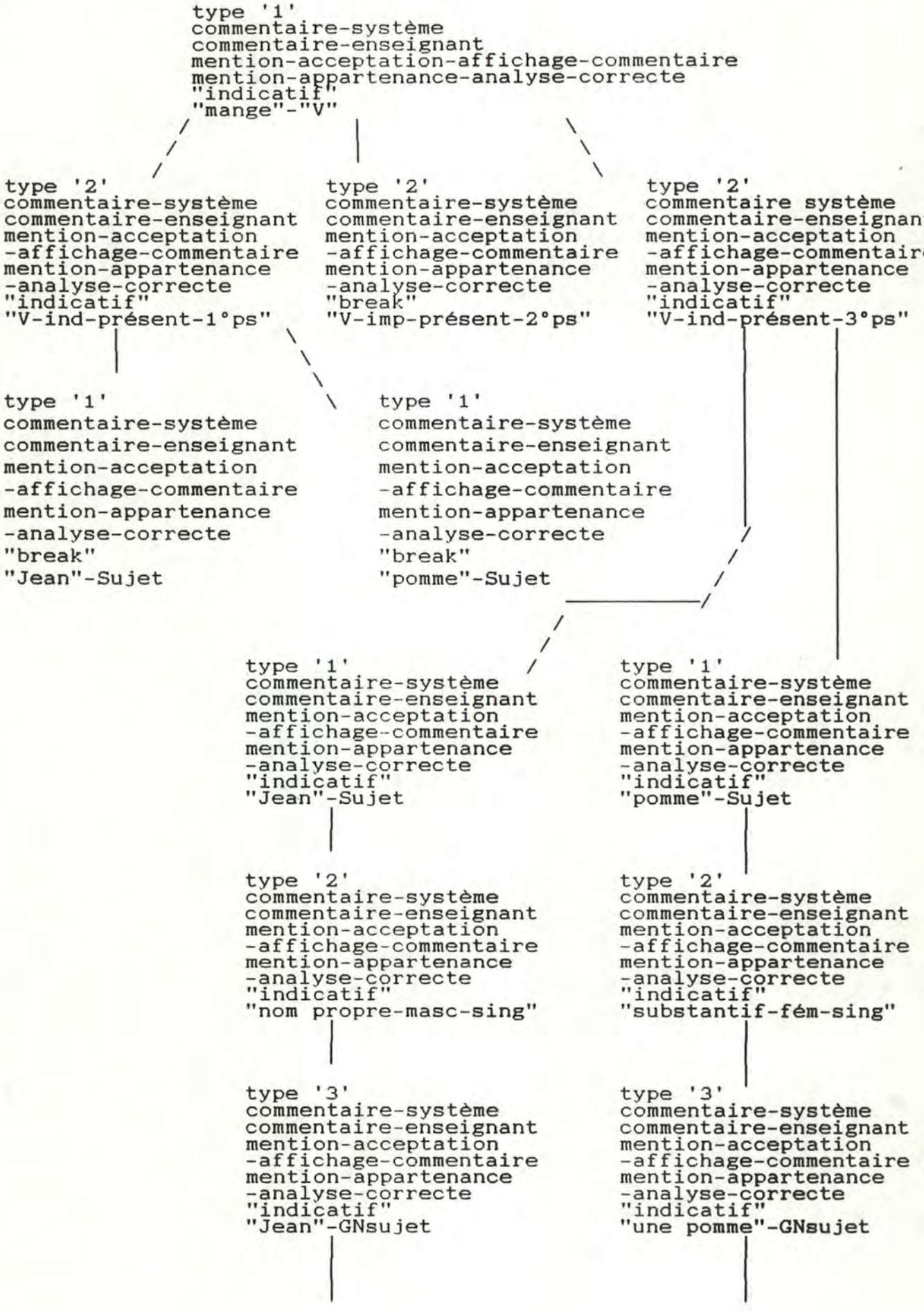
- similitude : ces structures sont toutes les deux un arbre de noeuds

- différences : - Une différence fondamentale entre ces deux structures de données est le contenu et l'utilisation de chacun de leurs noeuds :
  * un noeud de l'arbre d'analyse d'une phrase contient toutes les informations associées à une étape d'analyse de cette phrase. Elles sont utilisées par le système lors des exercices d'analyse et de traduction de la phrase pour établir un écran de dialogue avec l'élève.
  * un noeud du réseau d'écrans correspondant à l'analyse d'une phrase contient, lui, la description exacte de l'écran représentant une étape d'analyse de cette phrase, et pouvant s'afficher à un élève lors de son exercice d'analyse et de traduction de la phrase.

  - Une autre différence importante entre ces deux arbres est le fait que chaque noeud de l'arbre d'analyse est élaboré en partie par l'enseignant et en partie par le système. Ceci n'est pas le cas pour les noeuds du réseau d'écrans construits eux, entièrement par l'enseignant. Le système ne joue donc dans ce cas aucun rôle "productif" pouvant simplifier le travail de l'enseignant.

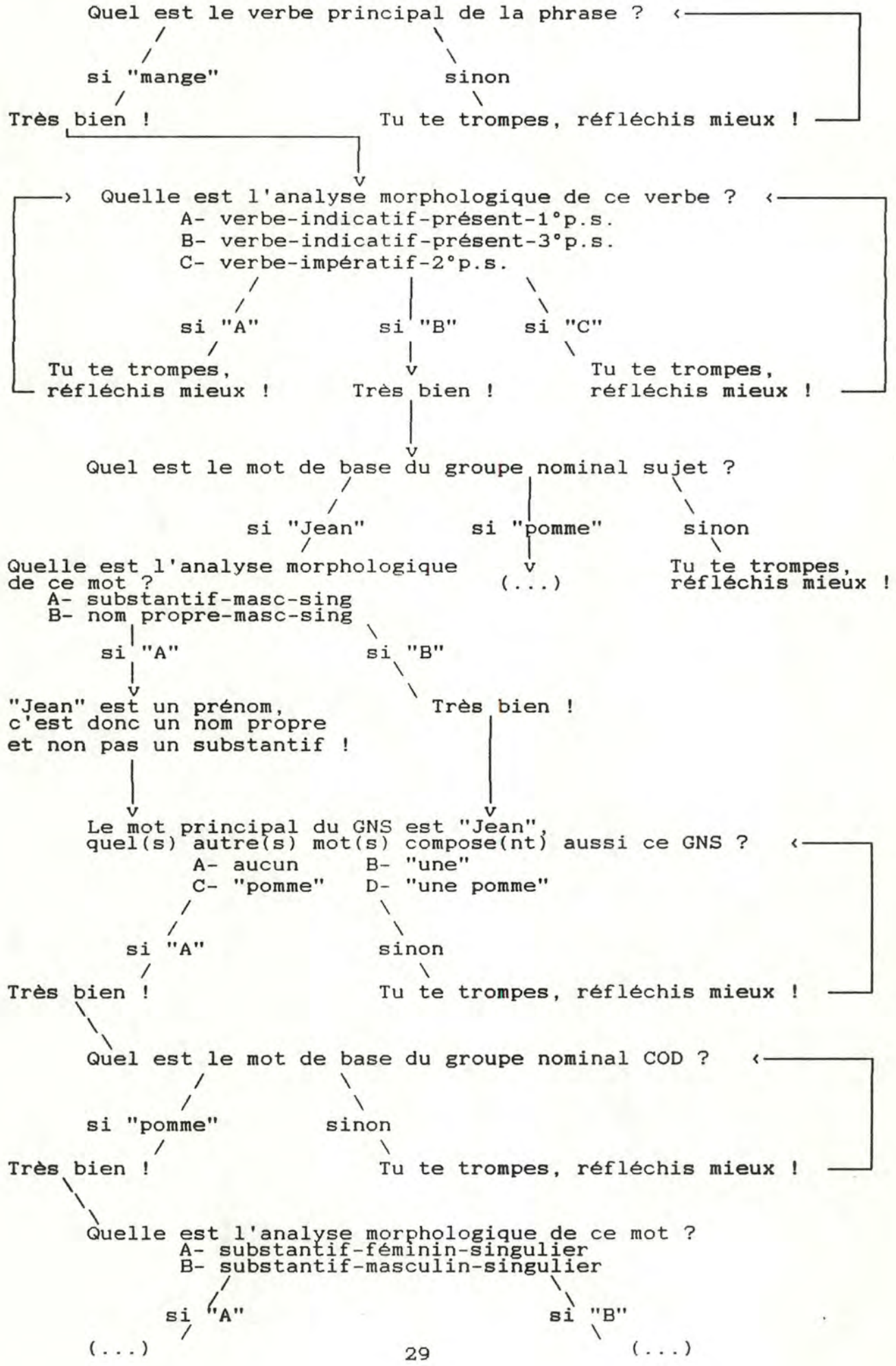
- illustration : analyse et traduction de la phrase suivante "Jean mange une pomme".

ARBRE D'ANALYSE




type '1'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"pomme"-COD
type '2'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"substantif-fém-sing"
type '3'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"une pomme"-GNcod
type '1'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"une"-déterminant
type '2'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"art-ind-fém-sing"
type '4'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
'traduction(s)'
type '1'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"une"-déterminant
type '2'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"indicatif"
"art-ind-fém-sing"
type '1'
commentaire-système
commentaire-enseignant
mention-acceptation
-affichage-commentaire
mention-appartenance
-analyse-correcte
"break"
"Jean"-COD

RESEAU D'ECRANS



5.1.2- Traitement automatique généré par le système

Le système va pouvoir tirer profit de son modèle de connaissances et l'exploiter afin de remplir une bonne partie de l'arbre d'analyse.

A l'origine, nous pensions que ce TRAITEMENT AUTOMATIQUE se déroulerait "off-line" plutôt que d'une manière "on-line", obligeant l'enseignant à rester bloqué longtemps, à attendre, devant son écran, les résultats du système.
Puisque le système travaillerait seul, sans nécessiter la présence de l'enseignant, ce dernier pourrait même demander au système de traiter "off-line" plusieurs textes spécifiques ou même tous les textes connus du système (éventuellement avec priorité à certains). Grâce à ses connaissances, le système remplirait, alors, au maximum, les arbres d'analyse de chaque phrase de ces textes traités. L'enseignant aurait, quand même, toujours la possibilité de stopper le travail du système quand il le désire, soit après le traitement d'un texte, soit pendant le traitement d'un texte et après ou pendant la considération d'une phrase de ce texte. Lors d'une demande d'interruption, ce que le système a déjà réalisé serait enregistré. Il serait alors possible à l'enseignant d'en prendre connaissance : pour chaque texte traité, à chaque phrase traitée, analysée automatiquement, serait associé le résultat du travail du système.

Cette analyse automatique "off-line" était prévue en pensant que le système contiendrait un analyseur morphologique. Mais, à présent que nous savons que ceci ne sera pas concrétisé dans notre travail, et que nous travaillerons sur des textes déjà analysés morphologiquement par le L.A.S.L.A. de l'Université de Liège, que devient ce traitement automatique "off-line" ?
Il contiendra tout traitement que le système peut réaliser seul de manière à faire progresser l'analyse de chaque phrase de chaque texte. Mais quels sont ces traitements automatiques ?
Seule la vérification des compléments manuels de l'enseignant, concernant les analyses morphologiques et syntaxiques, fera partie de ce traitement automatique : vérification de la cohérence des analyses syntaxiques et morphologiques d'une phrase. Si le système détecte des incohérences quant aux compléments manuels de l'analyse d'une phrase, ceux-ci devront être revus et corrigés par l'enseignant.

Nous savons qu'il n'existe aucun analyseur syntaxique automatique travaillant d'une manière totalement non-interactive, il faudra donc aider le système si l'on veut qu'il nous aide aussi, en trouvant cette analyse syntaxique. De plus, aucune analyse morphologique n'interviendra non plus lors de ce traitement automatique, puisque nous disposerons de textes déjà analysés lexicalement par le L.A.S.L.A. de l'Université de Liège.
Mentionnons, également, qu'une phrase ne pourra faire l'objet de ce traitement automatique que si l'état d'analyse dans lequel elle se trouve le permet.
La possibilité de demander au système de traiter automatiquement plusieurs textes latins, ainsi que le fait de pouvoir stopper n'importe quand ce traitement automatique resteront maintenus.
Ce traitement automatique vaut-il encore la peine d'être réalisé "off-line" ?
Seul son temps de réponse, lors de l'exécution du logiciel, nous renseignera !

5.1.3- Interventions de l'enseignant

Comme nous l'avons déjà expliqué, grâce à ses connaissances, le système peut générer automatiquement certaines informations relatives à l'analyse et à la traduction d'une phrase latine. Le travail de l'enseignant est donc allégé de ces besognes. Mais les connaissances du système ne lui permettent pas d'analyser toujours complètement et correctement une phrase latine. L'intervention de l'enseignant est dès lors nécessaire. Mais, celle-ci se limitera aux compléments des données générées automatiquement par le système.
Notre espoir est que le système soit capable de générer un grand nombre d'informations (étapes et messages) représentant l'arbre d'analyse d'une phrase et que le rôle de l'enseignant en soit réduit.

La tâche du professeur concernant les compléments de l'analyse d'une phrase est la suivante : confirmation ou infirmation des résultats établis par le système; à chaque intervention, gestion d'un message explicatif, et ajout d'informations inconnues du système, mais lui étant indispensables pour gérer correctement la "phase élève" . Ces interventions permettent de compléter l'arbre d'analyse de la phrase qui contient, alors, pratiquement toutes les informations nécessaires

au système pour gérer le dialogue qu'il aura avec l'élève, lors d'un exercice.

Nous avons décrit précédemment, dans la section 5.1.1, la démarche générale d'analyse d'une phrase. Nous pouvons, à présent, définir ce déroulement en considérant les étapes selon l'ordre "en profondeur d'abord" et "de gauche à droite". Donc, suivant cet ordre de considération de mots et suites de mots dans la phrase et de traitements à y affecter, le processus d'analyse permettant à l'enseignant de compléter l'arbre d'analyse est le suivant :

1- étape correspondant à un noeud de type '1' :
choix de l'élément de base caractérisant l'ensemble de mots à traiter à ce niveau de l'analyse selon l'ordre "en profondeur d'abord" et "de gauche à droite"
+ spécification du type de cette étape ("indicatif" ou "break")
+ possibilité d'introduction d'un commentaire spécifique à ce choix
+ introduction de la mention d'acceptation de l'affichage du commentaire à l'élève

2- étape correspondant à un noeud de type '2' :
analyse morphologique de ce mot : catégorie et caractéristiques
+ spécification du type de cette étape ("indicatif" ou "break")
+ possibilité d'introduction d'un commentaire spécifique à ce choix d'analyse morphologique
+ introduction de la mention d'acceptation de l'affichage du commentaire à l'élève

3- étape correspondant à un noeud de type '3' :
introduction des mots de la phrase correspondant au symbole catégoriel syntaxique à traiter à ce moment de l'analyse, selon l'ordre "en profondeur d'abord" et "de gauche à droite"
+ spécification du type de cette étape ("indicatif" ou "break")
+ possibilité d'introduction d'un commentaire spécifique à ce choix de découpe syntaxique
+ introduction de la mention d'acceptation de l'affichage du commentaire à l'élève

4- étape correspondant à un noeud d'analyse de type '2' :
analyse morphologique d'un mot appartenant à l'ensemble de mots venant d'être sélectionné, conformément au symbole catégoriel

syntaxique considéré
+ spécification du type de cette étape ("indicatif" ou "break")
+ possibilité d'introduction d'un commentaire spécifique à ce choix d'analyse morphologique
+ introduction de la mention d'acceptation de l'affichage du commentaire à l'élève
+ boucle de détail (goto '4') tant que tous les mots de cet ensemble de mots n'auront pas été analysés morphologiquement

5- Boucle de détails ('goto '1') tant que tous les mots de la phrase n'auront pas été complètement traités

6- étape correspondant à un noeud de type '4' :
traitement de l'admission de chaque traduction possible de chaque mot de la phrase afin de donner à celle-ci une traduction globale selon un certain sens
+ spécification du type de cette étape ("indicatif" ou "break")
+ possibilité d'introduction d'un commentaire spécifique à cette sélection
+ introduction de la mention d'acceptation de l'affichage du commentaire à l'élève
+ boucle (goto '6') s'il existe divers sens de traduction pour la phrase.

Nous remarquons, entre autres, dans cette démarche, l'importance des COMMENTAIRES associés à chaque noeud, correspondant, chacun, à une étape de l'analyse. Ces commentaires trouvent leur utilité auprès de l'élève. En effet, ils permettent au système d'établir un dialogue avec l'élève, de manière à aider ou encourager celui-ci dans son travail.
Notons que, lorsque l'élève introduit sa réponse correspondant à une étape d'analyse de la phrase, celle-ci peut être
soit 1- correcte
soit 2- possible mais non correcte dans le cas envisagé
(ex. un mot est morphologiquement correct pour être le verbe principal (= c'est un verbe), mais dans le cas envisagé, ce n'est pas le verbe principal)
soit 3- incorrecte par manque d'informations ou par informations erronées
(ex. un mot est morphologiquement incorrect pour être le verbe principal : c'est, par exemple, un adjectif)

Dans chaque cas, pour chaque noeud associé à une étape possible de l'analyse, l'enseignant a, nous le savons, la possibilité d'introduire un commentaire spécifique, ayant pour but de s'afficher à la place de celui généré automatiquement par le système :

si '1' : ce commentaire expliquera à l'élève pourquoi ce choix est correct et lui donnera éventuellement des informations supplémentaires quant à cette étape : caractéristiques d'un mot , ...

si '2' : ce commentaire expliquera à l'élève pourquoi ce choix est possible mais incorrect dans le contexte envisagé

si '3' : ce commentaire expliquera à l'élève pourquoi ce choix est incorrect, quel que soit le contexte envisagé.

Si l'enseignant n'associe lui-même aucun commentaire à un noeud correspondant à une étape d'analyse, le message généré par le système, grâce à ses connaissances, pourra s'afficher.

Le fait que le système puisse générer lui-même un commentaire explicatif pour chaque étape d'analyse est très important. En effet, cela simplifie encore le travail de l'enseignant qui n'a, alors, à introduire, lui-même, qu'un nombre limité de messages, puisqu'il existe toujours un commentaire "par défaut", celui du système.

Il faut, cependant, préciser que l'enseignant est parfois pratiquement obligé d'introduire lui-même un message pour certaines étapes d'analyse. Cette quasi-nécessité est due au fait que le système ne peut expliquer les erreurs de l'élève qu'en termes de ses propres connaissances. Dès lors, il peut quelquefois ne pas comprendre pourquoi l'élève a commis telle faute, car il ne possède pas les informations adéquates. Dans ce cas, bien souvent, l'enseignant, lui, connaît la raison de cette erreur.

ex. - intervention de la sémantique

- concernant l'analyse morphologique d'un mot, si le système possédait l'analyseur morphologique, il pourrait expliquer pourquoi, par exemple, tel mot ne peut pas être un verbe : radical, terminaisons, ... Mais comme le système ne contient pas cet analyseur, dans le cadre de notre travail, il ne peut fournir ces explications. Dans ce cas, il serait fort utile que l'enseignant introduise lui-même un commentaire plus détaillé.

Indiquons, dès à présent, que l'enseignant peut, également, lors de son analyse, associer un commentaire au texte analysé et à la phrase traitée. Cette association s'effectuera de la même manière que pour un noeud associé à une étape d'analyse. La seule différence réside dans le fait qu'aucun commentaire ne sera généré automatiquement par le système si l'enseignant n'associe aucun message à un texte ou une phrase.

L'objectif essentiel de notre travail n'est pas basé sur la gestion de ces commentaires. Nous nous contenterons donc, pour l'instant, d'un quelconque interface permettant à l'enseignant d'associer un message à un texte, une phrase ou un noeud correspondant à une étape d'analyse. Il sera, bien sûr, toujours envisageable de développer cette gestion des commentaires d'une manière plus élaborée par la suite .

Revenons-en, à présent,au traitement des COMPLEMENTS MANUELS DE L'ANALYSE D'UNE PHRASE apportés par l'enseignant.
La tâche de l'enseignant consiste à compléter les étapes d'analyse selon l'ordre "en profondeur d'abord" et de "gauche à droite". Ce travail aboutira à l'élaboration de l'arbre d'analyse complet de la phrase traitée.
Pour chaque noeud associé à une étape d'analyse, le système propose à l'enseignant les informations correspondantes qu'il a pu déterminer, lui-même, à partir de ces connaissances. Ces informations dépendront, bien entendu, du type de noeud envisagé.
S'il s'agit d'un noeud de type '1', suivant le type du groupe de mots à traiter à ce moment de l'analyse, le système en déduit la catégorie spécifiant l'élément de base. Par exemple, s'il s'agit d'un groupe nominal, l'élément de base est, en général, de la catégorie "substantif" ou "pronom". S'il s'agissait d'un groupe verbal, il faudrait considérer la catégorie "verbe". D'après les analyses morphologiques qu'il connait, le système peut alors détecter et proposer à l'enseignant les mots-candidats correspondants.
S'il s'agit d'un noeud de type '2', d'après la fonction du mot à analyser, le système sélectionne et propose à l'enseignant les analyses morphologiques possibles.
Pour un noeud de type '3' ou de type '4', le système est aussi capable d'exploiter ses connaissances pour proposer à l'enseignant ses résultats concernant l'étape d'analyse considérée.

A chaque proposition du système, un dialogue s'établit entre ce système et l'enseignant pour que l'ensemble des informations caractérisant chaque noeud d'analyse soit complet. L'enseignant intervient notamment pour préciser le type de cette étape d'analyse : s'il l'estime comme étant un noeud "break" ou "indicatif". Il a également la possibilité d'associer un commentaire spécifique à cette étape d'analyse. Et, il doit finalement indiquer s'il accepte que le commentaire s'affiche ou pas à l'élève, lors de son exercice.

Il est bien entendu que seuls sont considérés, pour une étape supplémentaire, les noeuds correspondants à une étape de l'analyse possible et acceptée, c'est-à-dire qualifiés d'"indicatif".

Remarquons que, pour les noeuds de type '3', le dialogue avec l'enseignant comporte éventuellement une étape supplémentaire. Ceci est dû au fait que le système ne contient pas assez de données pour effectuer toujours correctement l'analyse syntaxique. L'élaboration de cette découpe fait alors l'objet d'un échange d'informations entre l'enseignant et le système. Le système essaie, d'abord, d'établir seul l'analyse syntaxique du morceau de la phrase considérée (il propose à l'enseignant l'ensemble des mots susceptibles d'en faire partie). Puis, le professeur a alors la possibilité de "corriger" cette découpe proposée par le système et éventuellement incorrecte, dû à son manque de connaissances. Ce dialogue système-enseignant aboutit finalement à une analyse syntaxique exacte du morceau de la phrase.

La dernière étape visant à compléter l'arbre d'analyse d'une phrase est la gestion de la traduction de celle-ci. La tâche de l'enseignant, dans cette étape, en plus des compléments généraux définis précédemment, consiste à accepter des traductions françaises proposées par le système pour chaque mot de la phrase. En acceptant des traductions pour chacun des mots de la phrase, l'enseignant est supposé accepter chaque combinaison possible entre ces ensembles de traductions admises pour chaque mot, comme étant une traduction admise pour toute la phrase. Il est beaucoup plus facile, en effet, de gérer les traductions possibles d'une phrase en considérant qu'une traduction de la phrase est représentée par l'apposition des traductions de chaque

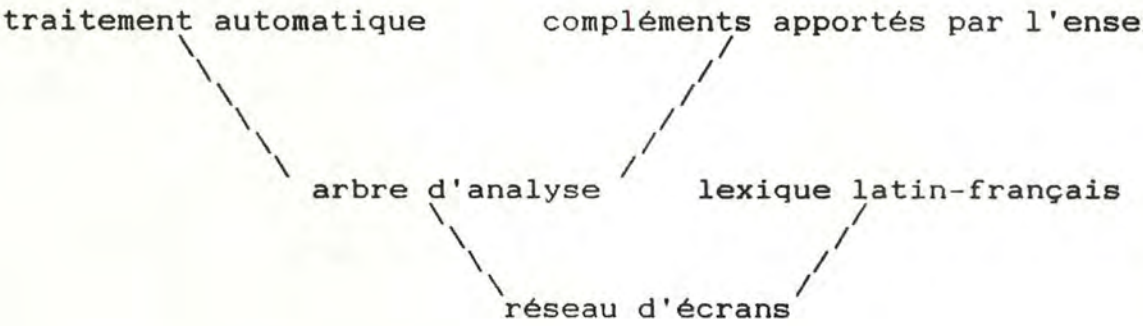
mot de cette phrase. Vouloir traduire la phrase latine par un ensemble de phrases françaises complètes est beaucoup plus compliqué, car la gestion des tournures correctes d'une phrase française est relativement complexe.
A partir de son lexique, le système propose à l'enseignant, une à une, toutes les traductions possibles de chaque mot de la phrase. Dès lors, pour chaque traduction française proposée, l'enseignant indique s'il accepte cette traduction possible comme étant admise pour ce mot, dans le contexte de la phrase et selon un certain sens.
Après avoir considéré tous les mots de la phrase et associé les compléments généraux à cette étape de traduction, l'enseignant a la possibilité d'indiquer s'il existe un autre sens suivant lequel interprêter et traduire cette phrase latine. Dans le cas positif, tout le processus de sélection des traductions possibles de chaque mot de la phrase recommence, afin d'en admettre certaines correspondant à un autre sens de traduction pour la phrase entière. Dans le cas négatif, l'analyse complète de la phrase latine est terminée.

Remarque :

Si ce n'est pas la première fois que l'analyse de cette phrase est complétée, les informations introduites, lors de précédents compléments sont affichées, au fur et à mesure des étapes d'analyse. L'enseignant peut alors maintenir ces choix ou les modifier.

5.1.4- Génération d'exercices d'analyse et de traduction de textes latins

Les arbres d'analyse remplis partiellement automatiquement par le système et complétés par l'enseignant, ainsi que le lexique latin-français constituent l'ensemble des informations nécessaires au système pour élaborer tous les dialogues s'établissant avec les élèves, lors de la résolution des exercices d'analyse et de traduction de ces textes latins.
La création d'occurences d'un exercice d'analyse et de traduction d'une phrase latine, représentées par un réseau d'écrans, peut donc être schématisée de la manière suivante :



Remarquons que, pour chaque exercice préparé, le système n'exploite qu'un certain nombre de noeuds appartenant à l'arbre d'analyse préparé. Seuls certains noeuds du réseau d'écrans correspondant à l'exercice sont générés. Cet ensemble de noeuds équivaut à une seule occurence de l'exercice, il s'agit, en fait, du dialogue précis établi entre l'élève et la machine lors de la résolution de cet exercice. De plus, il faut savoir que la génération de ces noeuds du réseau d'écrans n'est effectuée par le système que lors de la résolution de l'exercice par l'élève et non pas lors de la phase de préparation.

Un élément important de cette élaboration d'exercices d'analyse et de traduction latines est le LEXIQUE LATIN-FRANCAIS. Nous nous contenterons, pour l'instant, d'un lexique "figé", sur lequel l'enseignant n'a aucun pouvoir de modifications quant au nombre et à la forme des mots latins. Ceci n'est pas contraignant dans le cadre de notre travail, puisque l'ajout et la modification de formes latines dans le lexique n'apporteraient aucun renseignement supplémentaire au système. En effet, étant donné qu'il ne possède que les analyses morphologiques des formes latines en provenance du L.A.S.L.A. de l'Université de Liège, le

système ne saurait pas traiter lui-même seul une nouvelle forme latine introduite par l'enseignant dans le lexique.

5.2- Soumission aux élèves des exercices d'analyse et de traduction de textes latins

Lorsque la phase de préparation d'exercices d'analyse et de traduction de textes latins est terminée, ces exercices peuvent être proposés à des élèves.

Sachons que l'enseignant a la possibilité de savoir si tel ou tel texte est prêt à faire l'objet d'un exercice d'analyse et de traduction à proposer aux élèves. Cela signifie qu'il peut savoir si les arbres d'analyse de chaque phrase de ce texte sont complets. Pour cela, il peut visionner l'ETAT DE PREPARATION d'un texte latin, c'est-à-dire l'état de préparation de chaque phrase de ce texte. Si un texte n'est pas prêt, cela signifie qu'au moins une de ses phrases n'est pas prête : arbre d'analyse non complété par l'enseignant, incohérences détectées lors des compléments apportés par l'enseignant, ...

Un exercice d'analyse et de traduction d'un texte latin ne peut être proposé aux élèves que s'il a été complètement préparé. Un dialogue s'établit alors entre l'élève et l'ordinateur, tout au long de la résolution de l'exercice.
Détaillons, à présent, le scénario d'un exercice d'analyse et de traduction de textes latins soumis à un élève.

--› Analyse et traduction d'un texte latin :

Après avoir déterminé le texte sur lequel il veut travailler, l'élève le visionne et il a la possibilité d'en voir le commentaire associé, s'il existe. Etant donné qu'il va travailler sur un texte, phrase par phrase, n'analysant qu'une phrase à la fois, l'élève choisit, alors, la phrase de ce texte qu'il veut analyser et traduire.

--› Analyse et traduction d'une phrase latine

Lorsque l'élève a déterminé la phrase qu'il veut analyser, après l'avoir visionnée, sa tâche consiste alors à essayer d'en établir l'analyse et la traduction.

A quelque moment de l'analyse que ce soit, l'élève a toujours la possibilité de revoir le texte d'où provient la phrase qu'il est en train d'analyser, de voir le commentaire de ce texte, s'il existe, ou de voir le commentaire de la phrase traitée, s'il existe.
L'élève peut également interrompre son travail dès qu'il le désire, et non pas seulement après le traitement complet d'une phrase.
Lorsque l'analyse d'une phrase est terminée, l'élève peut alors sélectionner une autre phrase de ce texte à analyser ou un autre texte à traiter.

L'élève établit son analyse de la phrase selon un raisonnement semblable à celui suivant lequel l'enseignant a, lui, travaillé pour compléter l'arbre d'analyse associé à cette phrase. C'est-à-dire que les étapes d'analyse élaborées par l'élève sont du même genre que celles préparées par l'enseignant. Normalement, chaque étape d'analyse effectuée par l'élève correspond à un noeud de l'arbre d'analyse de préparation de la phrase. Dès lors, une analyse complète d'une phrase, introduite par l'élève, est associée à une suite de noeuds, un chemin dans l'arbre de préparation représentant toutes les étapes possibles de l'analyse de cette phrase.

A chaque étape d'analyse d'une phrase, l'élève a la possibilité d'indiquer qu'il pense s'être trompé dans sa démarche et qu'il désire "remonter" à l'étape précédente, au noeud supérieur.
Rappelons que si l'élève se trompe dans une étape de son analyse, quelle qu'elle soit, il n'en est pas obligatoirement averti. Le professeur aura choisi de qualifier le noeud correspondant de "break" ou d'"indicatif". Autrement dit, l'élève sera averti de son erreur et ne pourra pas continuer son analyse en suivant cette voie ou, au contraire, l'élève ne sera pas prévenu de sa faute et il pourra continuer dans cette voie d'analyse, en espérant que, par la suite, il se rende compte lui-même de son erreur.

Souvenons-nous, aussi, que nous avons supposer que l'analyse d'une phrase se déroule en considérant les mots à traiter et les étapes y associées selon l'ordre "en profondeur d'abord" et "de gauche à droite".

Dès lors, le processus général via lequel l'élève établit son analyse est semblable à celui suivi par l'enseignant, lors de la préparation :

1- étape correspondant aux informations d'un noeud de type '1' : choix de l'élément de base caractérisant l'ensemble de mots correspondant au symbole catégoriel à considérer à ce moment de l'analyse, en respectant l'ordre "en profondeur d'abord" et "de gauche à droite".

2- étape correspondant aux informations d'un noeud de type '2' : analyse morphologique d'une forme latine choisie lors de l'étape précédente : catégorie grammaticale et caractéristiques.

3- étape correspondant aux informations d'un noeud de type '3' : détermination de tous les mots appartenant à l'ensemble des mots associé au symbole catégoriel à traiter à ce moment de l'analyse, en respectant l'ordre "en profondeur" d'abord et "de gauche à droite", tout en sachant que l'élément principal de cet ensemble vient d'être déterminé et analysé.

4- étape correspondant aux informations d'un noeud de type '2' : analyse morphologique d'un mot appartenant à l'ensemble de mots venant d'être sélectionné, conformément au symbole catégoriel syntaxique considéré
+ boucle de détail (goto '4') tant que tous les mots de cet ensemble de mots n'auront pas été analysés morphologiquement par l'élève.

5- Boucle de détails ('goto '1') tant que tous les mots de la phrase n'auront pas été complètement traités par l'élève.

6- étape correspondant aux informations contenues dans le lexique et à celles d'un noeud d'analyse de type '4' :
introduction d'une traduction possible pour chaque mot de la phrase, selon l'analyse venant d'être établie, afin de donner à la phrase une traduction globale selon un certain sens.
Remarque :
Faut-il demander à l'élève de connaître toutes les traductions possibles de chaque mot, telles qu'elles sont contenues dans le lexique ?
Cela ne semble pas très réaliste. Une seule traduction pour chaque mot paraît suffisant de manière à obtenir une traduction

pour la phrase entière. Mais si, au contraire, il paraît intéressant de demander à l'élève toutes les traductions de chaque mot, telles qu'elles sont enregistrées dans le lexique, l'élève introduirait, alors, autant de traductions qu'il connaît. Le système lui indiquerait lesquelles sont correctes et, finalement, il lui donnerait, éventuellement, les autres traductions possibles qu'il ne connaît pas.

Un des problèmes résidant dans la réalisation du dialogue entre le système et l'élève, lors de ces étapes d'analyse, est de savoir s'il est préférable que ce dialogue s'effectue en termes de symbole catégoriel ou plutôt en termes de question directement en rapport avec les mots de la phrase :

ex. phrase = " Jean mange une pomme "

--› " quel est le C.O.D. ? "

--› " que mange Jean ? "

L'apport pédagogique est très différent dans les deux cas. Une solution n'est donc pas vraiment meilleure que l'autre, puisqu'elles ont toutes les deux un objectif différent !

De plus, dans le cas où le symbole catégoriel est envisagé, un second problème survient : l'élève doit-il introduire lui-même ce symbole ou pas ?

Nous ne pouvons pas résoudre seuls ces problèmes, ils relèvent d'une décision prise par des professeurs de latin.

Dans le cadre de notre travail, nous avons décidé, d'une part, de traiter les symboles catégoriels syntaxiques plutôt que les mots de la phrase, et, d'autre part, de ne pas demander à l'élève d'introduire ces symboles.

Ce choix étant relativement arbitraire, il pourra bien entendu être remis en question par la suite. Remarquons qu'il est très possible que les enseignants n'aient pas un point de vue unanime concernant ce problème. Il serait, alors, peut-être intéressant que chacun d'eux puisse décider du type de questions à poser à l'élève. Pour cela, il suffirait que l'enseignant indique son choix, lors de la phase de préparation, de manière à ce que le système adapte, en conséquence, son dialogue avec l'élève.

Rappelons que, quelle que soit la réponse fournie par l'élève à une quelconque étape d'analyse, en général, un commentaire explicatif adéquat s'affichera : ce sera le commentaire introduit par l'enseignant, lors de la phase de préparation, et associé à cette étape d'analyse ou le commentaire généré par le

système. Cet affichage se réalisera sauf si l'enseignant a précisé qu'à cette étape d'analyse, ne doit s'afficher aucun commentaire à l'élève.

Nous avons choisi de considérer le déroulement de l'analyse établie par l'élève selon l'ordre "en profondeur d'abord" et "de gauche à droite", concernant le suivi des étapes d'analyse et la considération des mots. Mais, il serait très intéressant d'ôter cette hypothèse et d'accorder davantage de libertés à l'élève. Aucun ordre ne lui serait imposé concernant le déroulement de l'analyse d'une phrase, il pourrait, alors, choisir, selon son gré, le suivi des étapes à effectuer de manière à aboutir à l'analyse complète de la phrase.
Nous pouvons même envisager de donner à l'élève la possibilité de choisir entre cette "liberté" et l'assistance du système. Un débutant pourrait faire ses exercices en se soumettant au déroulement de l'analyse imposé par le système. Puis, prenant de l'assurance, il déciderait d'analyser les phrases en choisissant lui-même l'ordre dans lequel il effectuerait ses étapes.

Dans l'état actuel des choses, le scénario décrivant le déroulement de l'analyse d'une phrase respecte l'ordre "en profondeur d'abord" et "de gauche à droite".
Pour chaque étape d'analyse, le système questionne l'élève en fonction du type de noeud y associé. Suite à la réponse de l'élève, le système peut déterminer le noeud de l'arbre d'analyse de préparation y correspondant. D'après les informations contenues dans ce noeud, le système sait si l'affichage du commentaire associé à cette étape d'analyse est permis ou pas. Dans le cas positif, le message adéquat apparaît. Il s'agit soit de celui que l'enseignant a introduit lors de la préparation, soit, si le commentaire du professeur n'existe pas, de celui généré par le système. Puis, d'après le type du noeud correspondant à cette étape ("indicatif" ou "break"), l'élève peut continuer l'analyse ou avoir une autre chance de réponse à cette étape d'analyse. Rappelons, également, que l'élève a toujours la possibilité de remonter à l'étape d'analyse précédente.

A propos des étapes d'analyse correspondant, chacune, à la détermination de l'analyse morphologique d'une quelconque forme latine, le cas particulier où la réponse de l'élève ne correspond pas exactement à l'une de celles prévues dans l'arbre de

préparation n'est pas à négliger. Il serait intéressant, dans ces situations, que le système fournisse automatiquement un commentaire "significatif". C'est-à-dire que, grâce à ses informations morphologiques, le système pourrait détecter quelles sont les caractéristiques incorrectes données par l'élève et celles exactes. Le commentaire à afficher pourrait alors être adapté en conséquence, de manière à fournir à l'élève une aide ou au moins un renseignement indicatif.

Concernant la dernière étape de l'analyse complète de la phrase, tout comme c'est le cas pour l'enseignant, l'élève choisit une ou plusieurs traduction(s) pour chaque mot composant la phrase, afin de donner à celle-ci une traduction globale, selon un certain sens. Deux possibilités peuvent être envisagées pour permettre à l'élève d'associer, à chaque mot latin, sa ou ses traduction(s) française(s). Celles-ci pourraient être soit, introduites au clavier par l'élève, soit, sélectionnées parmi celles proposées par le système. Une solution n'est pas pédagogiquement meilleure que l'autre. C'est pourquoi il serait intéressant que l'enseignant puisse déterminer, lors de la phase de préparation, quelle solution adopter.
Lorsqu'il a choisi toutes les traductions valables pour chaque mot de la phrase, dans son contexte et selon son analyse, les réponses de l'élève sont soumis au système :
Si la mention d'acceptation d'affichage du commentaire associé à cette étape d'analyse le permet, le commentaire adéquat va s'afficher. Puis, si le type du noeud correspondant à cette étape est "break", l'élève choisit soit d'avoir une autre chance pour établir sa sélection, soit de remonter à l'étape d'analyse précédente.

S'il existe un autre sens de traduction pour cette phrase,
soit le système le renseigne à l'élève qui doit, alors, resélectionner les traductions des mots de la phrase, en fonction de cet autre sens de traduction
soit l'élève doit réfléchir lui-même pour savoir s'il existe un autre sens de traduction pour cette phrase.

Mais avant tout, il faut se demander si l'élève DOIT ou seulement PEUT introduire tous les sens de traduction d'une phrase ?
Si nous considérions une interaction entre les phrases d'un texte, demander à l'élève de trouver tous les sens de traduction

d'une phrase ne semble pas très justifié. Il devrait, au contraire, trouver la traduction de la phrase correspondant au contexte du texte.
Dans notre travail, nous avons posé l'hypothèse de la non dépendance des phrases les unes entre les autres. Dans ce cas, il ne semble pas nécessaire, non plus, d'exiger que l'élève trouve tous les sens de traduction d'une phrase. A partir du moment où il en trouve une, cela signifie qu'il a su comprendre cette phrase selon un certain sens.
Mais il est tout à fait possible qu'un enseignant désire que tous les sens de traduction d'une phrase soient trouvés par l'élève. Le professeur devrait, donc, avoir aussi la possibilité de mentionner ce choix, lors de la phase de préparation.

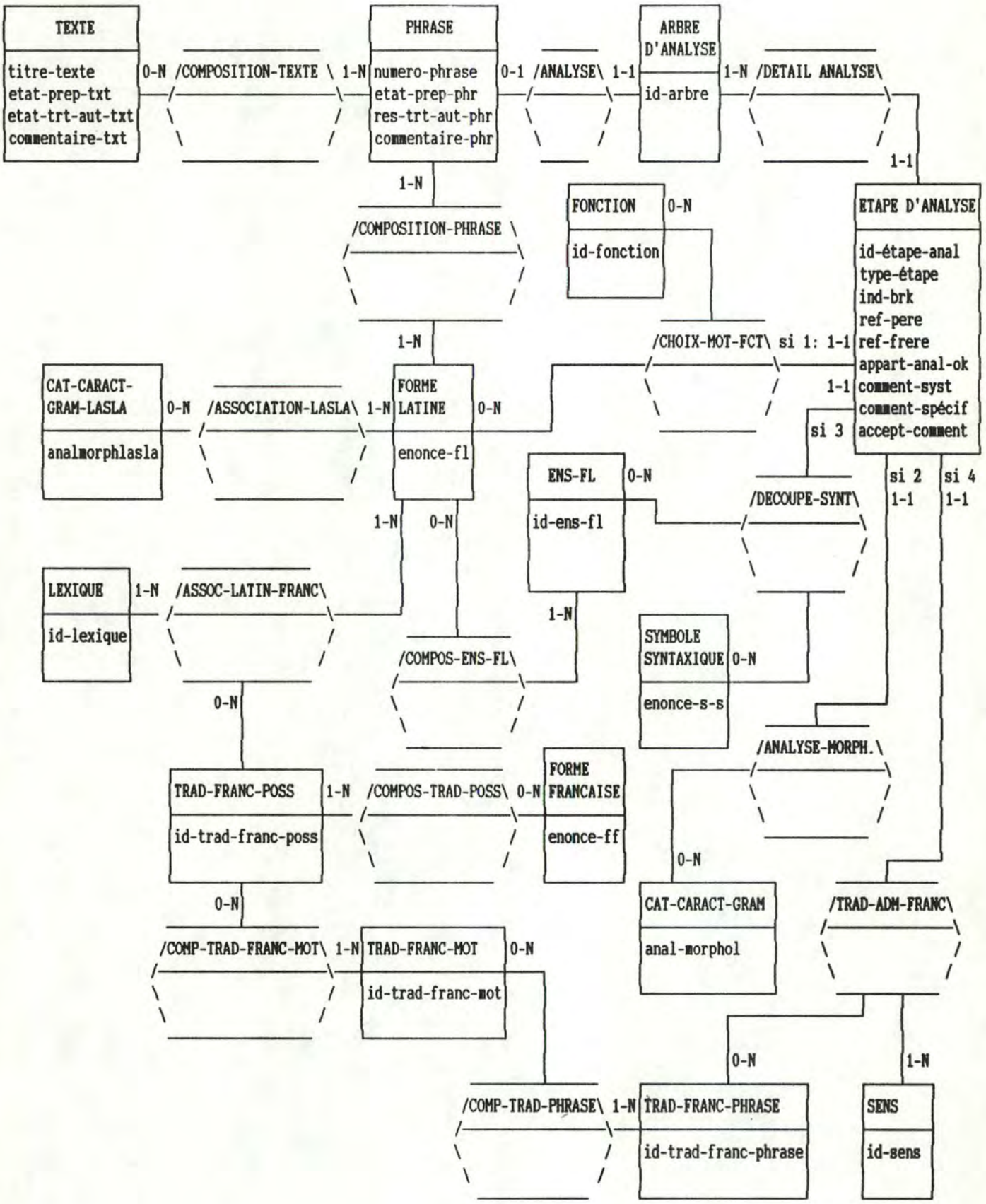
Quoi qu'il en soit, lorsque l'élève aura trouvé une ou plusieurs traduction(s) correcte(s) de la phrase qu'il traitait, l'analyse complète de celle-ci sera terminée et l'élève pourra, alors, choisir une autre phrase à analyser et traduire.

Remargue :

Si ce n'est pas la première fois que l'élève essaie d'analyser cette phrase, ses réponses introduites, lors de précédents exercices, sont affichées, au fur et à mesure des étapes d'analyse. L'élève peut alors maintenir ces choix ou les modifier.

CHAPITRE 6 : SCHEMA ENTITES-ASSOCIATIONS

6.1- Schéma



6.2- Descriptions des types d'entité

1- Type d'entité **TEXTE** :

Définition : une occurence du type d'entité TEXTE est un texte latin analysé morphologiquement par le LASLA de Liège. Ce texte et son analyse morphologique associée sont fournis par le L.A.S.L.A. de l'Université de Liège. Et le texte est l'objet d'un ensemble d'exercices d'analyse et traduction qui sont, d'abord, préparés par l'enseignant avec l'aide du système, puis soumis aux élèves.

Attributs :
- TITRE-TEXTE : titre du texte latin
  = chaîne de caractères
  propriétés : simple, élémentaire et obligatoire
- ETAT-PREP-TXT : état de préparation du texte
  = "prêt" ou "en préparation"
  propriétés : simple, élémentaire et obligatoire
- ETAT-TRT-AUT-TXT : état du traitement automatique du texte
  = "ok", "incohérences" ou "non traité automatiquement"
  propriétés : simple, élémentaire et obligatoire
- COMMENTAIRE-TXT : commentaire associé au texte
  = chaîne de caractères
  propriétés : simple, élémentaire et facultatif

Identifiant : TITRE-TEXTE

2- Type d'entité **PHRASE** :

Définition : une occurence du type d'entité PHRASE est une phrase latine appartenant à un ou éventuellement plusieurs texte(s), occurence(s) du type d'entité TEXTE.

Attributs :
- NUMERO-PHRASE : entier, numéro d'ordre de la phrase dans un texte latin, occurence du type d'entité TEXTE
  propriétés : simple, élémentaire et obligatoire
- ETAT-PREP-PHR : état de préparation de la phrase:
  '1' = "analyse du L.A.S.L.A."
  '2' = "compléments manuels de l'analyse en cours"

'3' = "compléments manuels de l'analyse terminés mais non vérifiés"
'4' = "compléments manuels de l'analyse incohérents"
'5' = "analyse prête"

propriétés : simple, élémentaire et obligatoire

- RES-TRT-AUT-PHR : résultat du traitement automatique de la phrase
= "ok", "incohérences" ou "non traitée"

propriétés : simple, élémentaire et obligatoire

- COMMENTAIRE-PHR : commentaire associé à la phrase
= chaîne de caractères

propriétés : simple, élémentaire et facultatif

Identifiants : (NUMERO-PHRASE, TITRE-TEXTE via COMPOSITION-TEXTE)

3- Type d'entité **FORME LATINE** :

Définition : une occurence du type d'entité FORME LATINE est une "unité lexicale" latine appartenant à une ou plusieurs phrase(s), occurence(s) du type d'entité PHRASE. Cette forme latine a été analysée morphologiquement par le L.A.S.L.A. de Liège.

Attributs : - ENONCE-FL : chaîne de caractères
propriétés : simple, élémentaire et obligatoire

Identifiant : ENONCE-FL

4- Type d'entité **LEXIQUE** :

Définition : une occurence du type d'entité LEXIQUE est un "dictionnaire" latin-français associant à chaque forme latine appartenant à une ou plusieurs phrase(s), occurence(s) du type d'entité PHRASE, une ou plusieurs traduction(s) française(s) possible(s).

Attributs : - ID-LEXIQUE : nom, identificateur du lexique
= chaîne de caractères

propriétés : simple, élémentaire et obligatoire

Identifiant : ID-LEXIQUE

5- Type d'entité **TRAD-FRANC-POSS** :

Définition : une occurence du type d'entité TRAD-FRANC-POSS est une suite de formes françaises, représentant une traduction française possible d'une forme latine.

Attributs : - ID-TRAD-FRANC-POSS : identificateur de cette traduction française
propriétés : simple, élémentaire et obligatoire

Identifiant : ID-TRAD-FRANC-POSS

6- Type d'entité **FORME FRANCAISE** :

Définition : une occurence du type d'entité FORME FRANCAISE est une "unité lexicale" française : que ce soit un mot tel qu'on le trouve dans un dictionnaire ou une forme conjugée ou accordée.

Attributs : - ENONCE-FF : chaîne de caractères
propriétés : simple, élémentaire et obligatoire

Identifiant : ENONCE-FF

7- Type d'entité **ARBRE D'ANALYSE** :

Définition : une occurence du type d'entité ARBRE D'ANALYSE est un arbre représentant toutes les informations associées aux étapes d'analyse possibles (correctes ou non) d'une phrase latine. Cet arbre a été établi par le système et complété par l'enseignant.

Attributs : - ID-ARBRE : identificateur de l'arbre
propriétés : simple, élémentaire et obligatoire

Identifiants : (ID-ARBRE, NUMERO-PHRASE via ANALYSE, TITRE-TEXTE via ANALYSE et COMPOSITION-TEXTE)

8- Type d'entité **ETAPE D'ANALYSE** :

Définition : une occurence du type d'entité ETAPE D'ANALYSE est un noeud de l'arbre complet de l'analyse d'une phrase. Ce noeud représente les informations associées à une étape de l'analyse d'une phrase, correcte ou pas.

Attributs : - ID-ETAPE-ANAL : identificateur du noeud associé à une étape d'analyse dans l'arbre d'analyse complet.
propriétés : simple, élémentaire et obligatoire

- TYPE-ETAPE : entier
  = '1' = "choix d'un mot correspondant à une fonction"
  '2' = "analyse morphologique d'une forme latine"
  '3' = "analyse syntaxique"
  (association d'un ensemble de mots à un symbole catégoriel syntaxique)
  '4' = "traductions admises de tous les mots de la phrase, selon un certain sens"

  propriétés : simple, élémentaire et obligatoire
- IND-BRK : "indicatif" ou "break"

  propriétés : simple, élémentaire et obligatoire
- REF-PERE : référence au noeud père de l'étape d'analyse dans l'arbre
  = identificateur du noeud père

  propriétés : simple, élémentaire et obligatoire
- REF-FRERE : référence au premier noeud frère de l'étape d'analyse dans l'arbre
  = identificateur du noeud frère

  propriétés : simple, élémentaire et obligatoire
- APPART-ANAL-OK : mention indiquant si le noeud correspond à une étape d'analyse appartenant à une analyse complète de la phrase, correcte ou pas

  propriétés : simple, élémentaire et obligatoire
- COMMENT-SYST : commentaire élaboré par le système et associé à ce noeud d'analyse

  propriétés : simple, élémentaire et obligatoire
- COMMENT-SPECIF : commentaire élaboré par l'enseignant et s'affichant à la place de celui du système, à l'élève

  propriétés : simple, élémentaire et facultatif
- ACCEPT-COMMENT : mention indiquant si le commentaire associé au noeud peut s'afficher ou pas à l'élève

  propriétés : simple, élémentaire et obligatoire

Identifiants : (ID-ETAPE-ANAL, ID-ARBRE via DETAIL ANALYSE)

9- Type d'entité **FONCTION** :

Définition : une occurence du type d'entité FONCTION est une fonction grammaticale pouvant être jouée par une forme latine dans une phrase ou dans un morceau de phrase latine

Attributs : - ID-FONCTION : identificateur de la fonction
= chaîne de caractères
propriétés : simple, élémentaire et obligatoire

Identifiant : ID-FONCTION

10- Type d'entité **CAT-CARACT-GRAM** :

Définition : une occurence du type d'entité CAT-CARACT-GRAM est une analyse morphologique d'une forme latine : catégorie grammaticale et caractérististiques y associées

Attributs : - ANAL-MORPHOL : informations grammaticales
propriétés : simple, décomposable et obligatoire

Identifiant : ANAL-MORPHOL

11- Type d'entité **CAT-CARACT-GRAM-LASLA** :

Définition : une occurence du type d'entité CAT-CARACT-GRAM-LASLA est une analyse morphologique d'une forme latine, élaborée par le L.A.S.L.A. de Liège : catégorie grammaticale et caractérististiques y associées

Attributs : - ANALMORPHLASLA : informations grammaticales codées sous forme de caractères (cfr. supra 7.1. Données initiales (p.64))
propriétés : simple, décomposable et obligatoire

Identifiant : ANALMORPHLASLA

12- Type d'entité **ENS-FL** :

Définition : une occurence du type d'entité ENS-FL est un ensemble de mots latins appartenant à une même phrase ou morceau de phrase latine, et correspondant à une découpe syntaxique de cette phrase ou morceau de phrase

Attributs : - ID-ENS-FL : identificateur de cet ensemble de formes latines

propriétés : simple, élémentaire et obligatoire

Identifiant : ID-ENS-FL

13- Type d'entité **SYMBOLE SYNTAXIQUE** :

Définition : une occurence du type d'entité SYMBOLE SYNTAXIQUE est un symbole syntaxique auquel pourra être associé un ensemble de mots latins, lors d'une découpe syntaxique d'une phrase ou d'un morceau de phrase latine

Attributs : - ENONCE-S-S : chaîne de caractères

propriétés : simple, élémentaire et obligatoire

Identifiant : ENONCE-S-S

Remargue :

Il ne faut pas confondre "SYMBOLE SYNTAXIQUE" et "CATEGORIE" ! Ces concepts sont différents : le premier est en rapport avec le point de vue syntaxique, tandis que l'autre s'attache au point de vue morphologique. Certaines occurences paraissent identiques, mais ne se rapportent pas à la même considération des choses :

ex. "SYMBOLE SYNTAXIQUE" : GN , V, GV , ADJ , SUJET, ...

"CATEGORIE" : SUBST , ADV , V , ADJ , ...

14- Type d'entité **TRAD-ADM-PHRASE** :

Définition : une occurence du type d'entité TRAD-ADM-PHRASE est un ensemble de traductions françaises admises pour une phrase latine, selon un certain sens.

Rappelons que nous considérons une traduction française d'une phrase latine comme étant l'apposition des traductions françaises de chaque mot latin de la phrase.

Attributs : - ID-TRAD-FRANC-PHRASE

propriétés : simple, élémentaire et obligatoire

Identificateur : (ID-TRAD-FRANC-PHRASE,

ID-SENS via TRAD-ADM-FRANC)

15- Type d'entité **SENS** :

Définition : une occurence du type d'entité SENS est une identification du sens, du contexte suivant lequel un ensemble de traductions admises d'une phrase doit être considéré.

Attributs : - ID-SENS = entier
propriétés : simple, élémentaire et obligatoire

Identifiant : ID-SENS

16- Type d'entité **TRAD-FRANC-MOT** :

Définition : une occurence du type d'entité TRAD-FRANC-MOT est un ensemble de traductions françaises d'une forme latine, admises selon un certain sens

Attributs : - ID-TRAD-FRANC-MOT = entier
propriétés : simple, élémentaire et obligatoire

Identifiant : (ID-TRAD-FRANC-MOT,
ID-TRAD-FRANC-PHRASE via COMP-TRAD-PHRASE,
ID-SENS via COMP-TRAD-PHRASE
et TRAD-ADM-FRANC)

6.3- Descriptions des types d'association

1- Type d'association **COMPOSITION-TEXTE** :

Types d'entité reliés : TEXTE et PHRASE
Définition : une occurence du type d'association COMPOSITION-TEXTE signifie qu'un texte latin existant est composé d'une ou plusieurs phrase(s) latine(s).
Rôles joués : - une occurence de TEXTE peut être composé d'une ou plusieurs occurence(s) de PHRASE
- une occurence de PHRASE "compose" une ou plusieurs occurence(s) de TEXTE
Connectivité : - (0-N) pour TEXTE
- (1-N) pour PHRASE
Identifiants : (NUMERO-PHRASE , TITRE-TEXTE)

2- Type d'association **COMPOSITION-PHRASE** :

Types d'entité reliés : PHRASE et FORME LATINE
Définition : une occurence du type d'association COMPOSITION-PHRASE signifie qu'une phrase latine existante est composée d'une ou plusieurs forme(s) latine(s)
Rôles joués : - une occurence de PHRASE est composée d'une ou plusieurs occurence(s) de FORME LATINE
- une occurence de FORME LATINE "compose" une ou plusieurs occurence(s) de PHRASE
Connectivité : - (1-N) pour PHRASE
- (1-N) pour FORME LATINE
Identifiants : (ENONCE-FL , NUMERO-PHRASE)

3- Type d'association **ASSOC-LATIN-FRANC** :

Types d'entité reliés : FORME LATINE, TRAD-FRANC-POSS et LEXIQUE
Définition : une occurence du type d'association ASSOC-LATIN-FRANC signifie qu'un lexique est composé d'associations reliant, chacune, une forme latine avec une traduction française possible
Rôles joués : - une occurence de LEXIQUE est composée d'une ou plusieurs association(s) reliant, chacune, une occurence de FORME LATINE à une occurence de TRAD-FRANC-POSS

- une occurence de FORME LATINE, associée à une occurence de TRAD-FRANC-POSS, appartient à une ou plusieurs occurence(s) de LEXIQUE
- une occurence de TRAD-FRANC-POSS peut être associée à une ou plusieurs occurence(s) de FORME LATINE et chaque association peut appartenir à une occurence de LEXIQUE

Connectivité : - (1-N) pour FORME LATINE
- (1-N) pour LEXIQUE
- (0-N) pour TRAD-FRANC-POSS

Identifiants : (ENONCE-FL, ID-TRAD-FRANC-POSS, ID-LEXIQUE)

4- Type d'association **ANALYSE** :

Types d'entité reliés : PHRASE et ARBRE D'ANALYSE

Définition : une occurence du type d'association ANALYSE signifie qu'à une phrase est associé un arbre d'analyse représentant toutes les informations correspondant aux étapes possibles de l'analyse complète de cette phrase. Cet arbre a été construit par le système et complété par l'enseignant.

Rôles joués : - une occurence de PHRASE peut être analysée par une seule occurence d'ARBRE D'ANALYSE
- une occurence d'ARBRE D'ANALYSE représente l'analyse d'une seule occurence de PHRASE

Connectivité : - (0-1) pour PHRASE
- (1-1) pour ARBRE D'ANALYSE

Identifiants : (NUMERO-PHRASE , TITRE-TEXTE via COMPOSITION-TEXTE , ID-ARBRE)

5- Type d'association **DETAIL ANALYSE** :

Types d'entité reliés : ARBRE D'ANALYSE et ETAPE D'ANALYSE

Définition : une occurence du type d'association DETAIL ANALYSE signifie qu'un arbre d'analyse est composé d'un graphe de noeuds correspondant, chacun, à une étape d'analyse

Rôles joués : - une occurence d'ARBRE D'ANALYSE est "détaillée" par une ou plusieurs occurence(s) d'ETAPE D'ANALYSE

- une occurence d'ETAPE D'ANALYSE "détaille", fait partie d'une seule occurence d'ARBRE D'ANALYSE

Connectivité : - (1-N) pour ARBRE D'ANALYSE
- (1-1) pour ETAPE D'ANALYSE

Identifiants : (ID-ARBRE, ID-ETAPE-ANAL)

6- Type d'association **CHOIX-MOT-FCT** :

Types d'entité reliés : ETAPE D'ANALYSE, FORME LATINE et FONCTION

Définition : une occurence du type d'association CHOIX-MOT-FCT signifie qu'un noeud d'analyse du type '1' est "caractérisé" par le choix d'une forme latine appartenant à une phrase ou un morceau de phrase latine et y représentant une certaine fonction

Rôles joués : - une occurence d'ETAPE D'ANALYSE du type '1' est spécifiée par le choix d'une occurence de FORME LATINE et d'une occurence de FONCTION
- une occurence de FORME LATINE, associée à une occurence de FONCTION, peut spécifier une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '1'
- une occurence de FONCTION, associée à une occurence de FORME LATINE, peut spécifier une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '1'

Connectivité : - (1-1) pour ETAPE D'ANALYSE
- (0-N) pour FORME LATINE
- (0-N) pour FONCTION

Identifiants : (ID-ETAPE-ANAL, ID-ARBRE via DETAIL ANALYSE, ENONCE-FL, ID-FONCTION)

7- Type d'association **ANALYSE-MORPH.** :

Types d'entité reliés : ETAPE D'ANALYSE, CAT-CARACT-GRAM

Définition : une occurence du type d'association ANALYSE-MORPH. signifie qu'un noeud d'analyse du type '2' est "caractérisé" par le choix d'une catégorie grammaticale et de caractéristiques y associées, représentant l'analyse morphologique d'une forme latine précédemment sélectionnée

Rôles joués : - une occurence d'ETAPE D'ANALYSE du type '2' est spécifiée par le choix d'une occurence de CAT-CARACT-GRAM
- une occurence de CAT-CARACT-GRAM peut spécifier une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '2'

Connectivité : - (1-1) pour ETAPE D'ANALYSE
- (0-N) pour CAT-CARACT-GRAM

Identifiants : (ID-ETAPE-ANAL, ID-ARBRE via DETAIL ANALYSE, ANAL-MORPHOL)

8- Type d'association **ASSOCIATION-LASLA** :

Types d'entité reliés : FORME LATINE, CAT-CARACT-GRAM-LASLA

Définition : une occurence du type d'association ASSOCIATION-LASLA signifie qu'à une forme latine, est associée une analyse morphologique : une catégorie grammaticale et une liste de caractéristiques y associée

Rôles joués : - une occurence de FORME LATINE est associée à une ou plusieurs occurence(s) de CAT-CARACT-GRAM-LASLA (= analyse morphologique établie par le L.A.S.L.A. de l'Université de Liège)
- une occurence de CAT-CARACT-GRAM-LASLA peut "analyser morphologiquement" une ou plusieurs occurence(s) de FORME LATINE

Connectivité : - (1-N) pour FORME LATINE
- (0-N) pour CAT-CARACT-GRAM-LASLA

Identifiants : (ENONCE-FL, ANALMORPHLASLA)

9- Type d'association **DECOUPE-SYNT** :

Types d'entité reliés : ETAPE D'ANALYSE, ENS-FL et SYMBOLE-SYNTAXIQUE

Définition : une occurence du type d'association DECOUPE-SYNT signifie qu'un noeud d'analyse du type '3' est "caractérisé" par le choix d'un ensemble de mots latins appartenant à une phrase ou un morceau de phrase et par un symbole syntaxique y associé

Rôles joués : - une occurence d'ETAPE D'ANALYSE du type '3' est spécifiée par le choix d'une occurence de ENS-FL et d'une occurence de SYMBOLE-SYNTAXIQUE

- une occurence de ENS-FL, associée à une occurence de SYMBOLE SYNTAXIQUE, peut spécifier une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '3'
- une occurence de SYMBOLE SYNTAXIQUE, associée à une occurence de ENS-FL, peut spécifier une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '3'

Connectivité : - (1-1) pour ETAPE D'ANALYSE
- (0-N) pour ENS-FL
- (0-N) pour SYMBOLE SYNTAXIQUE

Identifiants : (ID-ETAPE-ANAL, ID-ARBRE via DETAIL ANALYSE, ID-ENS-FL, ENONCE-S-S)

10- Type d'association **COMPOS-ENS-FL** :

Types d'entité reliés : FORME LATINE et ENS-FL

Définition : une occurence du type d'association COMPOS-ENS-FL signifie qu'un "ensemble de formes latines" est composé d'une ou plusieurs forme(s) latine(s)

Rôles joués : - une occurence de ENS-FL est composée d'une ou plusieurs occurence(s) de FORME LATINE
- une occurence de FORME LATINE peut composer une ou plusieurs occurence(s) de ENS-FL

Connectivité : - (1-N) pour ENS-FL
- (0-N) pour FORME LATINE

Identifiants : (ID-ENS-FL, ENONCE-FL)

11- Type d'association **TRAD-ADM-FRANC** :

Types d'entité reliés : ETAPE D'ANALYSE, TRAD-FRANC-PHRASE et SENS

Définition : une occurence du type d'association TRAD-ADM-FRANC signifie qu'un noeud d'analyse du type '4' est "caractérisé" par le choix d'un ensemble de traductions françaises admises pour la phrase analysée, selon un certain sens

Rôles joués : - une occurence d'ETAPE D'ANALYSE du type '4' est spécifiée par le choix d'une occurence de TRAD-FRANC-PHRASE et d'une occurence de SENS

- une occurence de TRAD-FRANC-PHRASE, associée à une occurence de SENS, peut spécifier une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '4'
- une occurence de SENS, associée à une occurence de TRAD-FRANC-PHRASE, spécifie une ou plusieurs occurence(s) d'ETAPE D'ANALYSE du type '4'

Connectivité :
- (1-1) pour ETAPE D'ANALYSE
- (0-N) pour TRAD-FRANC-PHRASE
- (1-N) pour SENS

Identifiants : (ID-ETAPE-ANAL, ID-ARBRE via DETAIL-ANALYSE, ID-SENS, ID-TRAD-FRANC-PHRASE)

12- Type d'association **COMP-TRAD-PHRASE** :

Types d'entité reliés : TRAD-FRANC-PHRASE, TRAD-FRANC-MOT

Définition : une occurence du type d'association COMP-TRAD-PHRASE signifie que l'on considère une traduction française d'une phrase latine comme l'apposition des traductions françaises de chaque mot de cette phrase

Rôles joués :
- une occurence de TRAD-FRANC-PHRASE est composée d'une ou plusieurs occurence(s) de TRAD-FRANC-MOT
- une occurence de TRAD-FRANC-MOT peut faire partie d'une ou plusieurs occurence(s) de TRAD-FRANC-PHRASE

Connectivité :
- (1-N) pour TRAD-FRANC-PHRASE
- (0-N) pour TRAD-FRANC-MOT

Identifiants : (ID-TRAD-FRANC-PHRASE, ID-TRAD-FRANC-MOT)

13- Type d'association **COMP-TRAD-FRANC-MOT** :

Types d'entité reliés : TRAD-FRANC-MOT, TRAD-FRANC-POSS

Définition : une occurence du type d'association COMP-TRAD-FRANC-MOT signifie qu'un ensemble de traductions françaises admises d'un mot latin, selon un certain sens est composé de traductions françaises possibles de cette forme latine

Rôles joués :
- une occurence de TRAD-FRANC-MOT est composée d'une ou plusieurs occurence(s) de TRAD-FRANC-POSS

- une occurence de TRAD-FRANC-POSS peut "composer" une ou plusieurs occurence(s) de TRAD-FRANC-MOT

Connectivité : - (1-N) pour TRAD-FRANC-MOT
- (0-N) pour TRAD-FRANC-POSS

Identifiants : (ID-TRAD-FRANC-POSS, ID-TRAD-FRANC-MOT)

14- Type d'association **COMPOS-TRAD-POSS** :

Types d'entité reliés : FORME FRANCAISE et TRAD-FRANC-POSS

Définition : une occurence du type d'association COMPOS-TRAD-POSS signifie qu'une traduction française possible d'un mot latin est "composée" d'au moins une forme française

Rôles joués : - une occurence de TRAD-FRANC-POSS est composée d'une ou plusieurs occurence(s) de FORME FRANCAISE
- une occurence de FORME FRANCAISE peut faire partie d'une ou plusieurs occurence(s) de TRAD-FRANC-POSS

Connectivité : - (1-N) pour TRAD-FRANC-POSS
- (0-N) pour FORME FRANCAISE

Identifiants : (ID-TRAD-FRANC-POSS, ENONCE-FF)

6.4- Contraintes d'intégrité

1- Etat de préparation d'un texte = "prêt"
‹==› pour CHAQUE phrase appartenant à ce texte :
état = "analyse prête"
= "analyse complétée manuellement, vérifiée et cohérente"

2- Etat de traitement automatique d'un texte
= "traité automatiquement"
‹==› il EXISTE au moins une phrase appartenant à ce texte, telle que l'état de cette phrase = "compléments manuels d'analyse terminés mais incohérents" ou "analyse prête"

3- Si l'état de préparation d'un texte = "prêt"
alors l'état de traitement automatique de ce texte
= "traité automatiquement"

4- Le résultat du traitement automatique d'une phrase ne peut valoir "incohérence"
QUE si l'état de préparation de cette phrase
= "compléments manuels d'analyse terminés mais incohérents"

5- Nous ne pouvons avoir à la fois qu'UNE seule des relations suivantes :
"choix-mot-fct", "analyse-morph.", "découpe-synt" ou "trad-adm-franc".
De plus l'association existante doit être cohérente avec le type du noeud associé à l'étape d'analyse à laquelle elle est reliée :
si "type-étape" = '1' alors "choix-mot-fct"
si "type-étape" = '2' alors "analyse-morph."
si "type-étape" = '3' alors "découpe-synt"
si "type-étape" = '4' alors "trad-adm-franc"

6- A une phrase latine, est associé un arbre d'analyse
‹==› état de cette phrase ‹› '1'

CHAPITRE 7 : IMPLEMENTATION

Détaillons, d'abord, comment nous avons exploité les données initiales en provenance de Liège, de manière à les adapter en vue de leur utilisation dans les programmes destinés à l'élaboration et l'exécution de leçons d'analyse et de traductions de textes latins. Nous spécifierons, ensuite, les données et procédures relatives, d'une part, à la création de leçons et, d'autre part, à leur exploitation auprès des élèves.

7.1:- Données initiales

Le L.A.S.L.A. de l'Université de Liège a mis à notre disposition deux fichiers décrivant respectivement le premier chapitre du "Livre Premier" et le premier chapitre du "Livre Deuxième" de l'oeuvre de César "La Guerre des Gaules" (les extraits en question sont repris en annexe 1). Ces deux fichiers sont chacun un "fichier text" dans lequel chaque ligne contient diverses informations concernant un mot du texte correspondant (et cela, dans l'ordre d'apparition des mots dans ce texte) :
la forme de ce mot dans le texte, le lemme y associé, la référence de ce mot dans ce texte, une analyse morphologique possible, ainsi que d'autres indications relatives à la tradition manuscrite et un code de ponctuation.
Ces informations sont mémorisées selon une certaine structure et un certain code :
chaque ligne du fichier contient 80 caractères, un renseignement est associé à une position particulière dans chaque ligne et chaque renseignement est codé sous forme de caractères.

Vous trouverez, à la page suivante, la description détaillée d'une de chaque ligne des fichiers.

Remarque :
Toute personne intéressée par le contenu de ces fichiers, se référera aux annexes 2 et 3.

position	signification
1	Code-carte
2-17	Lemme
18	Indice de lemme
19-38	Forme du texte
39-40	Code d'oeuvre Il s'agit de deux perforations permettant d'identifier l'oeuvre à laquelle le mot appartient.
41-53, 64,68-75	Référence du mot
41-43	Numéro du chapitre (001, 002 etc.)
44-47	Numéro du paragraphe ou du vers (0001, 0002 etc.)
48-50	Numéro d'ordre du mot dans le paragraphe ou dans le vers (001, 002 etc.)
51-53	Numéro d'ordre du mot dans la phrase (001, 002 etc.)
64	Code du rôle pour les oeuvres dramatiques
68-70	Indication éventuelle de la subdivision en livres
71-75	Numéro d'ordre du mot dans l'oeuvre (00001, 00002 etc.)
54-63	Analyse complète de la forme
65	Indications relatives à la tradition manuscrite
66	Ponctuation du texte
67	Colonne libre
76-80	Numéro d'ordre du mot dans l'index.

Les conventions adoptées à propos de "l'Analyse" d'une forme, c'est-à-dire les caractères renseignés à partir de la position 54 jusqu'à celle 66, sont mentionnées à la page suivante.

54 — Catégorie grammaticale

Substantif	1
Adjectif	7
Numéral	3
Adjectif pronom	4
Verbe	5
Adverbe	6
Préposition	7
Conjonction	8
Interjection	9

55 — Sous-catégorie, Degré, Voix

Substantif 1

1e déct	1
2e déct	2
3e déct	3
4e déct	4
5e déct.	5
Anomal	6
Déct gr.	7

Adjectif 7

	P	C	S
1e classe	1	A	J
2e cl cons	2	B	K
-er	3	C	L
-is	4	D	M
-imp	5	F	N
Anomal	6	F	O
Déct gr.	-		

Numéral 3

	P	C	S
Cardinal	1		
Ordinal	2	B	K
Distributif	3		
Multiplic	4		
Adv ord.	5	E	N
Adv mult	6		

Adjectif pronom 4

Personnel	1
Possessif	2
Réfléchi	3
Possessif réfléchi	4
Démonstratif	5
Relatif	6
Interrogatif	7
Indéfini	8

Verbe 5

	A	P	D	S	D
1e conj	1	A	J		
2e conj	2	B	K	S	
3e conj	3	C	L	T	
4e conj	4	D	M		
4e bis	5	E	N		
Anomal	6	F	O		

Adverbe 6

Relatif	6
Interrogatif	7
Négatif	8
Interrogatif négatif	9
Comparatif	12
Superlatif	11

Préposition 7

Type MECTIM	1

Conjonction 8

Coordination	1
Subordination	2

Interjection 9

56 — Cas, Personne, Nombre

	S	P
Nominatif	A	J
Vocatif	B	K
Accusatif	C	L
Génitif	D	M
Datif	E	N
Ablatif	F	O
Locatif	G	P
Indéclinable	Z	

Verbe 5

	S	P
1e	A	J
2e	B	K
3e	C	L

Cas régi 3, 4 ou 6

57 — Mode

Mode — du verbe subordonné régi par un pronom relatif ou interrogatif (cf. verbe)

Indicatif	1
Impératif	2
Subjonctif	3
Participe	4
Adj. verbal	5
Gérondif	6
Infinitif	7
Supin en UM	8
Supin en U	9

Mode — du verbe subordonné régi par un adverbe relatif ou interrogatif (cf. verbe)

Mode — du verbe subordonné régi par une conjonction de subordination (cf. verbe)

58 — Temps

Temps — du verbe subordonné régi par un pronom relatif ou interrogatif (cf. verbe)

Présent	1
Imparfait	2
Futur simple	3
Parfait	4
Plus que parfait	5
Futur antérieur	6
-us fui/fuerim/fuisse	7
-us fuerem/fuissem	8
-us fuero/urus fuisse	9

Temps — du verbe subordonné régi par un adverbe relatif ou interrogatif (cf. verbe)

Temps — du verbe subordonné régi par une conjonction de subordination (cf. verbe)

59 — Fonctions

Verbe principal	12
Verbe subord	11

60 — Emploi

Emploi subst.	1
adject.	2
adverb	6

61 — Genre - Tmèse

a) Genre

commun	1
féminin	2
masc. et fém.	3
masculin	4
masc. et neutre	5
neutre	6

b) Tmèse 12

65 — Tradition

Leçon contestée	1
Mot absent dans certains mss. et atteint par édit	2
Conjecture reçue	3
Lacune comblée	4
CRUX	5
Leçon rejetée	6
Mot présent dans certains mss. et rejeté par édit	7
Mot impossible à analyser en fonction d'une lacune	0

66 — Ponctuation

K	dernier mot du chap. du paragraphe ou du vers et de la phrase
S	dernier mot du paragraphe ou du vers et de la phrase
7	dernier mot de la phrase
11	dernier mot du chapitre et du paragraphe ou du vers
0	dernier mot du paragraphe ou du vers

CODE ALPHABETIQUE

	12	11	Zéro
1	A	J	/
2	B	K	S
3	C	L	T
4	D	M	U
5	E	N	V
6	F	O	W
7	G	P	X
8	H	Q	Y
9	I	R	Z

La lettre O sera notée Ø
Le zéro sera noté O
La conjonction du digit 1 et du zoning zéro est représentée non par une lettre mais par le signe /

62 - 63 — CODES DE SUBORDINATION

subjonctif seul complément ..... AA	qualiscumque ..... GD	quomodo (rel.) ..... PN
ablatif absolu ..... AD	quam (rel.) ..... GG	quomodo (int.) ..... PS
proposition infinitive ..... AG	quam (int.) ..... GK	quoniam ..... PX
	quamdiu (adv. int.) ..... GN	quoquo ..... RA
ac ..... AK	quamdiu C S ..... GS	quot (rel.) ..... RD
proinde ac si ..... AN	quamdudum ..... GX	quot (int.) ..... RG
perinde ac si ..... AS	quamlibet ..... HA	quotcumque ..... RK
... an ..... AX	quamobrem (rel.) ..... HD	quotiens (rel.) ..... RN
an ... an ..... BA	quamobrem (int.) ..... HG	quotiens (int.) ..... RS
... anne ..... BD	quamquam ..... HK	quotienscumque ..... RX
anteaquam ..... BG	quamvis ..... HN	quotquot ..... SA
antequam ..... BK	quando ADV ..... HS	quotus ..... SD
	quando C S ..... HX	quotuscumque (rel.) ..... SG
cum ..... BN	quandocumque ..... JA	quotusquisque ..... SK
cumcumque ..... BS	quandoque ..... JD	quousque ..... SN
cur ..... BX	quandoquidem ..... JG	
	quantopere (rel.) ..... JK	seu ..... SS
donec ..... CA	quantopere (int.) ..... JN	si ..... SX
dum ..... CD	quantulus (rel.) ..... JS	sicut ..... TA
dummodo ..... CG	quantulus (int.) ..... JX	simulac ..... TD
dumtaxat ..... CK	quantuluscumque ..... KA	sin ..... TG
	quantus (rel.) ..... KD	siquidem ..... TK
etiamsi ..... CN	quantus (int.) ..... KG	sive ..... TN
etsi ..... CS	quantuscumque ..... KK	
	quare (rel.) ..... KN	tametsi ..... TS
licet CX	quare (int.) ..... KS	tamquam ..... TX
	quasi ..... KX	tamquam si ..... WA
modo ..... DA	quatenus (rel.) ..... LA	
	quatenus (int.) ..... LD	ubi (rel.) ..... WD
ne C S ..... DD	quemadmodum (rel.) ..... LG	ubi (int.) ..... WG
ne ..... DG	quemadmodum (int.) ..... LK	ubi C S ..... WK
ne ... an ..... DK	qui (rel.) ..... LN	ubicumque ..... WN
ne ... anne ..... DN	qui (int.) ..... LS	unde (rel.) ..... WS
necne ..... DS	qui (abl. rel.) LX	unde (int.) ..... WX
nedum ..... DX	qui (abl. int.) ..... MA	undecumque ..... XA
neve, neu ..... EA	quia ..... MD	ut (adv. rel.) ..... XD
nl ..... ED	quicumque ..... MG	ut (adv. int.) ..... XG
nisi ..... EG	quin C S ..... MK	ut C S ..... XK
nonne ..... EK	quippe ..... MN	ut si ..... XN
num ..... EN	quis ..... MS	utcumque ..... XS
	quisnam ..... MX	uter (rel.) XX
postquam ..... ES	quisquis ..... NA	uter (int.) ..... YA
posteaquam ..... EX	quo (rel.) ..... ND	utercumque ..... YD
priusquam ..... FA	quo (int.) ..... NG	uti (adv. rel.) ..... YG
proquam ..... FD	quo ADV ..... NK	uti (adv. int.) ..... YK
prout ..... FG	quo C S ..... NN	uti C S ..... YN
	(non) quofd) ..... NS	utqui ..... YS
qua (rel.) ..... FK	quoad (adv. int.) ..... NX	utrum ... an ..... YX
qua (int.) ..... FN	quoad ..... PA	utrum ... necne ..... ZA
quacumque ..... FS	quocumque ..... PD	
qualis (rel.) ..... FX	quod ..... PG	velut ..... ZD
qualis (int.) ..... GA	quominus ..... PK	velut si ..... ZG

Mais seules certaines de ces informations nous intéressent dans le cadre de notre travail. Il s'agit des données nous permettant de reconstituer les phrases du texte latin analysé et de celles nous indiquant, outre la forme de chaque mot du texte, le lemme y associé et les analyses morphologiques possibles.

Remarquons que, par manque de données suffisantes, l'analyseur automatique du L.A.S.L.A. de l'Université de Liège ne parvient parfois pas à déterminer d'analyse morphologique ou de lemme pour certaines formes latines ou définit des renseignements incomplets. Dès lors, pour pallier l'inexistence de certaines informations nécessaires (ex. lemme), quelques compléments manuels ont été effectués grâce au programme GESTAM (cfr.supra). Mais rappelons que l'objectif de notre travail consiste en la l'indication de l'efficacité d'un système "semi-automatique". Par la suite, de tels ajouts ne seront plus nécessaires, étant donné notre intention d'automatiser, nous-mêmes, la création des analyses morphologiques de chaque mot d'un quelconque texte. Cela nous permettrait de devenir indépendants et de ne plus recourir au L.A.S.L.A. de Liège pour l'obtention de ces données.

A partir de ces renseignements, une structure de données plus adaptée à notre travail est élaborée grâce au programme **CONVERT2** (cfr. annexe 6). Celui-ci, à partir d'un fichier texte contenant des informations répondant à la structure des données spécifiée par le L.A.S.L.A. de l'Université de Liège, met à jour le fichier général **'b:fmotlat.am'** (cfr. annexe 4). Chaque record de ce fichier global est structuré de la manière suivante : chaque record contient un mot latin, une analyse morphologique possible, le lemme y associé et une traduction française possible (cfr.supra). Ces records sont triés selon l'ordre alphabétique des mots latins.
Un fichier "index" (**'b:index.am'**) est associé à 'b:fmotlat.am'. Chaque record de ce fichier contient un mot latin et la position de sa première analyse morphologique dans 'b:fmotlat.am'. Ces records sont également triés selon l'ordre alphabétique des mots latins et mis à jour par CONVERT2. Le contenu de ce fichier est mentionné en annexe 5.
Ce programme CONVERT2 est aussi responsable de la création de deux autres fichiers plus spécifiques. Ceux-ci, contrairement à ceux précédemment cités, ne sont pas "généraux", mais sont associés à un texte latin particulier.

Le premier (**'b:nomfichierlasla.fct'**) est un fichier "text" qui contient le texte latin, tel qu'il apparaît à l'écran. Ce qui signifie que, pour chaque texte, deux mots successifs sont séparés d'un et un seul blanc, un point termine chaque phrase, chaque nouvelle phrase commence en début de ligne et aucun mot n'est coupé en fin de phrase.
Le deuxième fichier (**'b:nomfichierlasla.dsc'**) contient, lui, des informations générales concernant le texte latin associé. Ces renseignements sont les suivants et sont modifiables en cours d'analyse :

- à propos du texte dans son entièreté :
  * le titre associé
  * l'état de préparation de l'analyse de ce texte
    ("prêt" ou "en préparation")
  * le résultat du traitement automatique vérifiant la cohérence de l'analyse de ce texte
    ("incohérence", "ok" ou "non encore traité")
  * un commentaire général
  * le nom du fichier contenant le texte
- à propos de chaque phrase du texte :
  * le numéro de cette phrase dans le texte
  * l'état de préparation de l'analyse de cette phrase
    ("analyse fournie par le LASLA de l'Université de Liège",
      "compléments manuels de l'analyse en cours",
      "compléments manuels de l'analyse finis mais non vérifiés",
      "compléments manuels de l'analyse incohérents"
      ou "analyse cohérente")
  * le résultat du traitement automatique vérifiant la cohérence de l'analyse de cette phrase
    ("cohérente", "incohérente" ou "non encore traitée")
  * un commentaire particulier associé à cette phrase
  * le nom du fichier contenant "l'analyse" de cette phrase

Ces informations, selon leur nature, sont mises à jour soit automatiquement, soit par l'enseignant.

Remarquons, d'une part, que nous avons préféré ne pas introduire les lemmes, les analyses morphologiques et les traductions françaises dans les fichiers contenant les textes latins. Cela aurait, en effet, alourdi la manipulation de ces données, étant donné le gonflement de chaque fichier et l'hétérogénéité des informations. Cela aurait, également, provoqué de

nombreuses redondances de renseignements. Nous avons, donc, préféré séparer les données selon leur utilité et créer un fichier contenant lemmes, analyses morphologiques et traductions françaises associés à chaque forme latine traitée.
Le problème se posant, alors, était le choix entre la possibilité d'élaborer un fichier GENERAL 'b:fmotlat.am', contenant les informations relatives à tous les textes traités ou la possibilité d'associer un fichier PARTICULIER 'b:fmotlat.am' à chaque texte considéré.
Etant donné notre situation actuelle dans laquelle nous ne disposons que de deux textes, nous avons choisi de créer UN fichier général 'b:fmotlat.am'. Celui-ci est donc valable pour les deux textes et sa manipulation est simplifiée grâce au fichier 'b:index.am'.
Il est certain que, par la suite, ce choix pourra être remis en question. En effet, le gonflement de ce fichier peut poser problème. D'un autre côté, il faudra également étudier le fait de savoir comment gérer les disquettes : faut-il associer une disquette à chaque élève ou une disquette à chaque texte ? Selon ce choix, il faudra traiter le fichier 'b:fmotlat.am' différemment.

Nous avons mentionné précédemment que notre fichier 'b:fmotlat.am' contient les lemmes et les analyses morphologiques possibles de chaque mot latin apparaissant dans les fichiers "convertis" décrivant des textes latins. Mais ce fichier contient également une traduction française possible pour chacun de ces mots. Cette considération lexicale est nécessaire car notre travail consiste à gérer l'analyse d'une phrase latine, mais également à élaborer sa traduction française comme étant, rappelons-le, l'apposition des traductions de chacun de ses mots.
Quelle solution adopter pour traiter un lexique latin-français ? Nous pourrions créer un fichier contenant, triées alphabétiquement, toutes les formes latines apparaissant dans les textes "convertis" et leurs traductions françaises possibles y associées. Mais, remarquons l'interdépendance entre la traduction d'un mot et son analyse morphologique ! Dès lors, créer un fichier lexique indépendamment de celui des analyses morphologiques exigerait une manipulation assez lourde et complexe, due aux interrelations mentionnées ci-dessus.

Pourquoi, alors, ne pas intégrer ces informations lexicales dans le fichier 'b:fmotlat.am' ? Il est certain que ceci gonfle ce fichier, mais le fichier index y associé en simplifie quand même la manipulation. Nous avons donc opté pour cette solution.

Pour l'instant, limitons-nous à une traduction française pour chaque analyse morphologique de chaque mot de ce fichier global. Une généralisation sera, sans aucun doute, utile par la suite. Celle-ci permettrait de faire intervenir davantage de sémantique pour pouvoir choisir une traduction parmi plusieurs possibles, en tenant compte du contexte grammatical de chaque mot dans la phrase. A ce moment-là, il faudra également penser, comme dans le cas de 'b:fmotlat.am', à la possibilité de créer et gérer un ou plusieurs lexiques, chose que nous ne réalisons pas actuellement, vu nos restrictions concernant le nombre de ces traductions.

Les fichiers en provenance du L.A.S.L.A. de l'Université de Liège ne contiennent aucune information lexicale. Le programme CONVERT2 ne peut donc associer automatiquement aucune traduction française possible aux mots latins. Dès lors, il nous a fallu introduire ces données manuellement, par l'intermédiaire du programme **GESTAM**, dont le code écrit en Turbo Pascal 4.0 se trouve en annexe 7. Celui-ci a été conçu pour permettre la mise à jour manuelle du fichier 'b:fmotlat.am' en ce qui concerne les traductions françaises, mais, également, les lemmes et les analyses morphologiques, éventuellement inexistants dans les fichiers originaux, en provenance du L.A.S.L.A.

Le programme **TESTS** (cfr. annexe 8) nous offre, finalement, la possibilité de visualiser le contenu de chaque fichier contenant un texte latin ('*.fct'), de chaque fichier décrivant un texte latin ('*.dsc'), ainsi que du fichier 'b:fmotlat.am' et de son index 'b:index.am'.

Toutes ces données étant disponibles, nous pouvons à présent définir les deux parties principales du didacticiel d'aide à l'analyse et à la traduction de textes latins. Premièrement, il s'agit de celle permettant à l'enseignant d'élaborer des leçons sur ces textes disponibles et, deuxièment, de celle destinée aux élèves profitant de ces leçons préparées à leur intention.

7.2:- Aide à l'élaboration de leçons

7.2.1- Fichier d'analyse d'une phrase latine : description

Le programme d'aide à l'élaboration de leçons, destiné au professeur, a comme objectif de créer ou mettre à jour "l'analyse" de chaque phrase latine traitée. Chaque "analyse", comme décrit dans les sections précédentes, est représentée par un arbre contenant un ensemble d'informations, correspondant aux étapes d'analyses possibles correctes ou non de la phrase associée.
Chaque arbre d'analyse, au cours du traitement, est représenté par une liste chaînée. Chaque maillon de cette liste est un record composé, d'une part, d'un ensemble d'informations décrivant une étape d'analyse et, d'autre part, de trois pointeurs pointant respectivement vers le "père", le "frère" et le "fils" de ce noeud.
Un fichier d'analyse ('**b:nomfichierlasla.numerophrase**') mémorisera ensuite ces données.
La structure d'un tel fichier est calquée sur celle de la liste chaînée. C'est-à-dire que chacun de ses records contient, d'une part, la description d'une étape d'analyse et, d'autre part, deux booléens. Ces deux booléens mentionnent l'existence d'un noeud fils et d'un noeud frère représentant également une étape d'analyse de la phrase associée.
Nous trouvons, donc, dans chaque record les informations suivantes, décrivant une étape d'analyse :

- la mention du type de l'étape d'analyse : indicatif ou break

- la mention de l'appartenance de ce noeud à une analyse correcte de la phrase

- le commentaire du système associé automatiquement à cette étape d'analyse

- le commentaire spécifique du professeur pouvant être ajouté, de manière à s'afficher à l'élève à la place de celui du système

- la mention d'acceptation de l'affichage du commentaire (spécifique ou celui du système) lors de la phase destinée à l'élève

- le type du noeud :

n1 = choix d'un mot correspondant à une fonction

n2 = analyse morphologique d'un mot

n3 = association d'un ensemble de mots à un symbole catégoriel syntaxique

n4 = traduction de la phrase entière

- selon le type du noeud, diverses informations sont mentionnées:

si n1 : - le mot choisi
- la fonction le qualifiant (sujet, verbe, cod ou cplt)

si n2 : - le lemme associé à l'analyse morphologique du mot choisi lors d'une étape n1
- la catégorie morphologique de ce mot (substantif, adjectif, numéral, adjectif pronom, verbe, adverbe, préposition, conjonction ou interjection)
- selon la catégorie morphologique du mot, diverses informations qualifient l'analyse, conformément aux renseignements du L.A.S.L.A. :

si substantif : déclinaison, cas, nombre et genre

si adjectif : classe, degré, cas, nombre et genre

si numéral : catégorie, degré, cas, nombre et genre

si adjectif pronom : catégorie, cas, nombre, genre et mode et temps du verbe subordonné, dans le cas d'un pronom relatif ou interrogatif

si verbe : conjugaison, voix, fonction, genre, temps, mode et, selon le mode, personne et nombre ou cas et nombre

si adverbe : catégorie et mode et temps du verbe subordonné, dans le cas d'un adverbe interrogatif ou relatif

si préposition : cas régi et mention de l'appartenance de cette préposition au type "mecum"

si conjonction : catégorie et mode et temps du verbe subordonné, dans le cas d'une conjonction de subordination

si n3 : - le symbole catégoriel syntaxique considéré (gnsujet, gncod ou gnabl)
- un tableau de booléens mentionnant l'appartenance de chaque mot de la phrase à la découpe syntaxique

Comme vous l'avez certainement remarqué, seules certaines fonctions (sujet, verbe, cod et cplt) et seuls certains symboles catégoriels syntaxiques (gnsujet, gncod et gnabl) sont considérés. Cela est dû au fait que les phrases latines peuvent avoir parfois une structure assez complexe. Or, étant donné notre intention d'élaborer un outil de démonstration "valable", certaines hypothèses ont été émises quant à la structure des phrases latines acceptées par notre système. Celles-ci doivent être composées d'un groupe nominal sujet non vide (GNs), d'un verbe principal, d'un groupe nominal COD non vide (GNcod) et d'un groupe nominal complément à l'ablatif non vide (GNcplt). Cela, sachant que

- un **GNs** peut comprendre les éléments suivants :
substantif(s)-nominatif
et/ou adjectif(s)-nominatif
et/ou adjectif(s) pronom(s) (personnel, réfléchi, démonstratif, interrogatif, indéfini) -nominatif

- un **VERBE** est soit une forme verbale simple, soit une forme verbale composée d'une forme verbale de "esse" et d'un participe passé

- un **GNcod** peut comprendre les éléments suivants :
substantif(s)-accusatif
et/ou adjectif(s)-accusatif
et/ou adjectif(s) pronom(s) (personnel, réfléchi, démonstratif, interrogatif, indéfini) -accusatif
et/ou préposition(s) régissant l'accusatif
et/ou "et"

- un **GNcplt** peut comprendre les éléments suivants :
substantif(s)-ablatif
et/ou adjectif(s)-ablatif
et/ou adjectif(s) pronom(s) (personnel, réfléchi, démonstratif, interrogatif, indéfini) -ablatif
et/ou préposition(s) régissant l'ablatif
et/ou "et"

Remarquons directement que, parmi les phrases mises à notre disposition, deux d'entre elles sont conformes à cette structure. Il s'agit de la deuxième et la troisième phrase du premier chapitre du "Livre Premier" de l'oeuvre de César "La Guerre des

Gaules". Elles pourront donc faire l'objet de la démonstration du programme.
Une généralisation de ce programme, en vue de pouvoir traiter des phrases latines de structure syntaxique quelconque est, bien évidemment, à envisager par la suite.

7.2.2- Spécifications des procédures

Outre l'aide à l'élaboration, proprement dite, de leçons concernant l'analyse et la traduction de phrases latines, le programme **LATPROF**, spécifiquement destiné à être utilisé par l'enseignant, offre diverses autres possibilités à exploiter par le professeur. Décrivons ces propositions offertes dès le menu principal par le système :

1. "LISTE TEXTES LATINS" : **LIST_TXT_LAT**

Argument : /
Précondition : /
Résultat : /
Postcondition : affichage à l'écran des noms des fichiers se trouvant sur la disquette dans le drive 'b' et possédant l'extension '.fct' (ces fichiers contiennent donc chacun un texte latin pouvant être analysé).

2. "VISUALISATION TEXTE LATIN" : **VISU_TXT_LAT**

Argument : namefcttxt
Précondition : "namefcttxt" est un string[14] représentant le nom d'un fichier 'b:*.fct' existant contenant un texte latin.
Résultat : /
Postconditions : affichage à l'écran du texte latin contenu dans le fichier portant le nom "namefcttxt".
Si ce texte est trop long que pour apparaître en un écran, la pression d'une touche au clavier permettra d'en voir la suite. De même, à la fin du texte, il suffira d'appuyer sur une touche au clavier pour retourner d'où on est venu (=menu).

3. "COMPLEMENTS ANALYSE ET TRADUCTION TEXTE LATIN" :
COMPLEMENTS

Argument : namefct

Préconditions : "namefct" est un string[14] représentant le nom d'un fichier 'b:*.fct' existant, contenant un texte latin.
Le fichier 'b:*.dsc' y associé doit également être présent sur la disquette du drive 'b', ainsi que les fichiers contenant les analyses des phrases déjà traitées et les fichiers généraux 'b:fmotlat.am' et 'b:index.am'.

Résultat : COMPLEMENTS offre la possibilité d'établir ou mettre à jour "l'Analyse" d'une ou plusieurs phrases de ce texte contenu dans le fichier portant le nom "namefct".

Postconditions : Le fichier 'b:*.dsc' sera mis à jour selon les modifications établies durant cette préparation, et les fichiers associés aux analyses des phrases traitées seront, eux aussi, mis à jour ou créés.

4. "TRAITEMENT AUTOMATIQUE : VERIFICATION COHERENCE ANALYSE TEXTE LATIN : **TRT_AUT_COH**

Cette procédure n'a pas été développée au sein de notre travail, faute de temps. Il serait, cependant, utile de l'élaborer, par la suite, de manière à pouvoir vérifier "off-line" la cohérence des compléments effectués par l'enseignant au niveau de l'analyse d'une ou plusieurs phrase(s).
Cette vérification pourrait avoir lieu pour une phrase particulière, pour plusieurs, pour toutes les phrases analysées d'un texte ou même pour toutes les phrases analysées de plusieurs textes.
Suite aux messages mentionnés par le système après ces vérifications et selon leur nature (simple avertissement : ex. "la phrase analysée ne comporte pas de verbe" ou mention d'incohérence : ex. "le mot sélectionné comme centre du GNsujet ne se retrouve pas dans l'ensemble des mots formant ce GNsujet"), l'enseignant devra alors agir en conséquence.

5. "LISTE RESULTATS TRAITEMENT AUTOMATIQUE" : **LIST_RES_TRT_AUT**

Argument : namefdt

Précondition : "namefdt" est un string[14] représentant le nom d'un fichier 'b:*.dsc' existant, contenant des informations décrivant le fichier associé 'b:*.fct'.

Résultat : /

Postconditions : affichage à l'écran des informations décrivant les résultats du traitement automatique vérifiant la cohérence de l'analyse de chaque phrase du texte latin 'b:*.fct'. Le résultat global est d'abord affiché (non encore traité, incohérence(s) ou correct), puis le résultat de chaque phrase, prise indépendamment, apparaît (non encore traitée, incohérence(s) ou correcte). Si ces informations ne peuvent s'afficher en un seul écran, la pression d'une touche au clavier permettra d'en voir la suite. De même, à la fin de l'affichage, il suffira de presser une touche au clavier pour retourner d'où on est venu (= du menu).

6. "VISUALISATION ETAT PREPARATION TEXTE LATIN" :
VISU_ETAT_PREP_TXT_LAT

Argument : namefdt

Précondition : "namefdt" est un string[14] représentant le nom d'un fichier 'b:*.dsc' existant, contenant des informations décrivant le fichier 'b:*.fct'.

Résultat : /

Postcondition : affichage à l'écran des informations décrivant les états de préparation de l'analyse de chaque phrase du texte latin 'b:*.fct'. L'état de préparation global est d'abord affiché (prêt ou en préparation), puis l'état de préparation de chaque phrase, prise indépendamment, apparaît (analyse fournie par le LASLA de Liège, compléments manuels en cours, compléments manuels finis mais non vérifiés, compléments manuels incohérents ou analyse cohérente). Si ces informations ne peuvent pas s'afficher en un seul

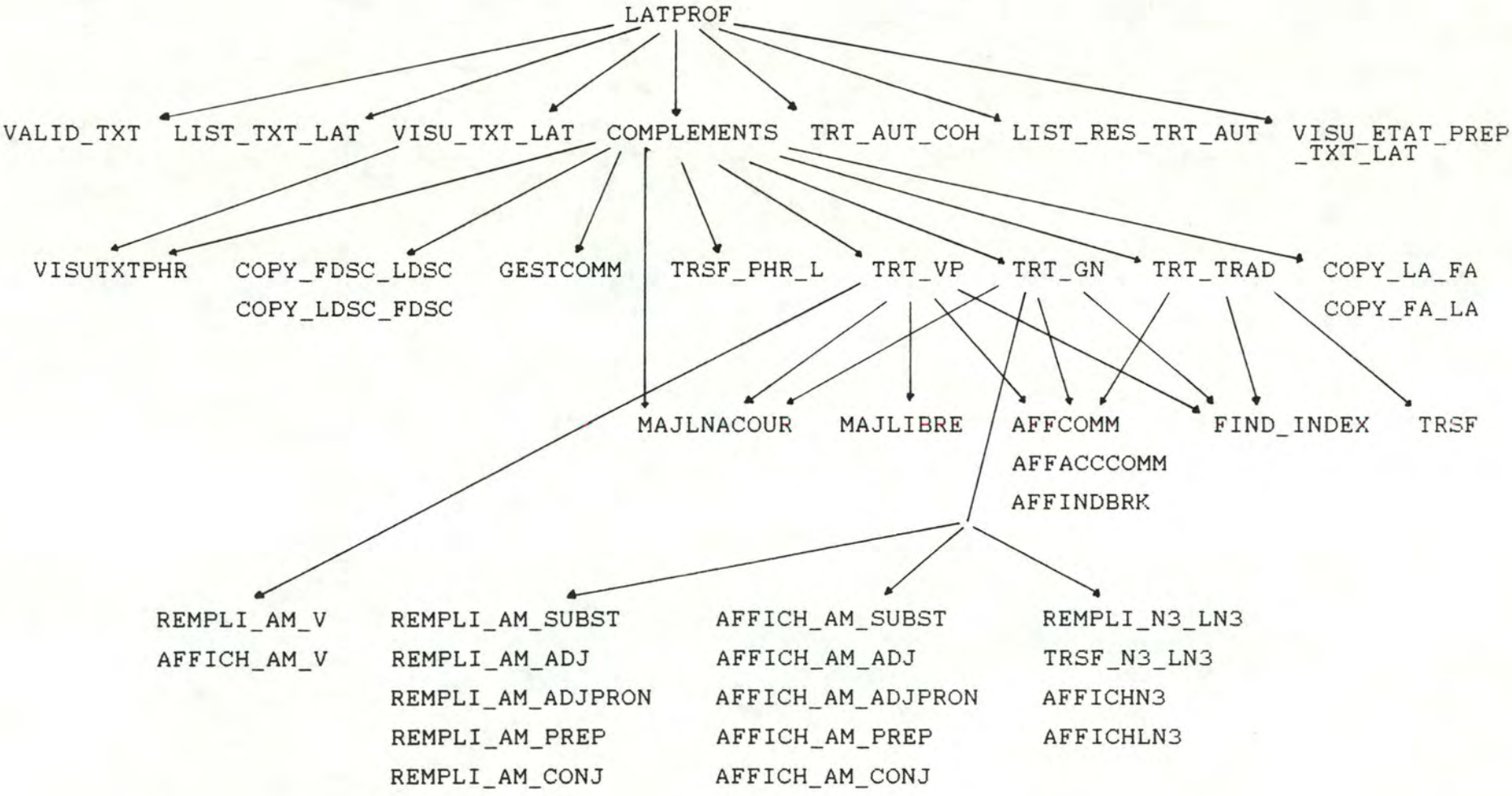
écran, la pression d'une touche au clavier permettra d'en voir la suite. De même, à la fin de l'affichage, il suffira de presser une touche au clavier pour retourner d'où on est venu, c'est-à-dire ici du menu.

Le graphe reliant les procédures utilisées, dans cette partie du didacticiel, se trouve à la page suivante. La relation "appelle" structurant ce graphe est définie de la manière suivante :

Proc1 ----------› Proc2 (appelle)

signifie que dans le corps de la procédure Proc1, apparaît au moins une instruction d'appel de la procédure Proc2.

Après ce schéma, vous trouverez les spécifications des procédures de plus bas niveau que celles précitées.


LATPROF
VALID_TXT
LIST_TXT_LAT
VISU_TXT_LAT
COMPLEMENTS
TRT_AUT_COH
LIST_RES_TRT_AUT
VISU_ETAT_PREP
_TXT_LAT
VISUTXTPHR
COPY_FDSC_LDSC
COPY_LDSC_FDSC
GESTCOMM
TRSF_PHR_L
TRT_VP
TRT_GN
TRT_TRAD
COPY_LA_FA
COPY_FA_LA
MAJLNACOUR
MAJLIBRE
AFFCOMM
AFFACCCOMM
AFFINDBRK
FIND_INDEX
TRSF
REMPLI_AM_V
AFFICH_AM_V
REMPLI_AM_SUBST
REMPLI_AM_ADJ
REMPLI_AM_ADJPRON
REMPLI_AM_PREP
REMPLI_AM_CONJ
AFFICH_AM_SUBST
AFFICH_AM_ADJ
AFFICH_AM_ADJPRON
AFFICH_AM_PREP
AFFICH_AM_CONJ
REMPLI_N3_LN3
TRSF_N3_LN3
AFFICHN3
AFFICHLN3

7. **VALID_TXT**

Argument : nametxt

Précondition : "nametxt" est un string[14].

Résultat : VALID_TXT prend la valeur 'true' si 'b:nametxt.fct' correspond au nom d'un fichier existant contenant un texte latin, sinon VALID_TXT prend la valeur 'false'.

Postcondition : /

8. **VISUTXTPHR**

Arguments : namefct
nphrase
title
miny
maxy

Préconditions : "namefct" est un string[14] représentant le nom d'un fichier 'b:*.fct' existant, contenant un texte latin.
"nphrase" est un entier appartenant à l'intervalle [0..nombremaximumdephrasedutexte].
"title" est un string[60].
"miny" et "maxy" sont des entiers appartenant à l'intervalle [1..25].

Résultat : result

Postconditions : SI "nphrase" = '0'
ALORS VISUTXTPHR affiche à l'écran entre les lignes "miny" et "maxy" d'abord "title", puis toutes les phrases contenues dans le fichier portant le nom "namefct" et "result" vaudra 'true'.
Si le texte est trop long que pour apparaître en un écran limité par "miny" et "maxy", un message en "may + 2" demandera la pression d'une touche pour en voir la suite.
SINON VISUTXTPHR affiche à l'écran entre les lignes "miny" et "maxy" d'abord "title", puis la phrase portant le numéro "nphrase" dans le texte contenu dans le fichier portant le nom "namefct". Si la phrase est trop longue que pour apparaître en un écran limité par "miny" et

"maxy", un message en "maxy + 2" demandera la pression d'une touche pour en voir la suite. "result" vaudra 'true' si la phrase portant le numéro "nphrase" existe dans le fichier "namefct", sinon "result" vaudra 'false'.

9. **TRSF_PHR_L**

Arguments : namefct
nphrase

Préconditions : "namefct" est un string[14] représentant le nom d'un fichier 'b:*.fct' existant, contenant un texte latin.
"nphrase" est un entier appartenant à l'intervalle [1..nombremaximumdephrasedutexte].

Résultat : fmcp

Postconditions : TRSF_PHR_L recopie la phrase portant le numéro "nphrase", dans le texte latin mémorisé dans le fichier "namefct", dans une liste chaînée plus facilement manipulable. Chaque maillon de la liste contient un mot de la phrase transformé sous forme de string[20] 'complet'.
"fmcp" contient un pointeur pointant vers le premier maillon de la chaîne.

10. **COPY_FDSC_LDSC**

Argument : namefdt

Préconditions : "namefdt" est un string[14] représentant un fichier 'b:*.dsc' existant, contenant des informations décrivant le fichier associé 'b:*.fct'.

Résultat : firstld

Postcondition : COPY_FDSC_LDSC copie le contenu du fichier décrivant un texte latin et portant le nom "namefdt" dans une liste chaînée plus facilement manipulable. "firstld" contiendra alors un pointeur vers le premier maillon de la chaîne.

11. **COPY_LDSC_FDSC**

Arguments : namefdt
firstld

Préconditions : "namefdt" est un string[14].
"firstld" est un pointeur pointant vers le premier maillon de la liste chaînée contenant les informations décrivant le fichier associé à "namefdt", 'b:*.fct'.

Résultat : namefdt

Postcondition : COPY_LDSC_FDSC copie dans le fichier décrivant un texte latin et portant le nom "namefdt", les informations y associées et contenues dans la liste chaînée dont le premier maillon est accessible par "firstld".

12. **COPY_LA_FA**

Arguments : nomfichanal
firstlna

Préconditions : "nomfichanal" est un string[14].
"firstlna" est un pointeur pointant vers le premier maillon de la liste chaînée contenant les informations décrivant l'Analyse d'une phrase latine déjà (partiellement) traitée.

Résultat : nomfichanal

Postcondition : COPY_LA_FA copie dans le fichier décrivant l'Analyse d'une phrase latine et portant le nom "nomfichanal", les informations y associées et contenues dans la liste chaînée dont le premier maillon est accessible par "firstlna".

13. **COPY_FA_LA**

Argument : nomfichanal

Préconditions : "nomfichanal" est un string[14] représentant le fichier existant contenant l'Analyse d'une phrase latine.

Résultat : firstlna

Postconditions : COPY_FA_LA copie le contenu du fichier décrivant l'Analyse d'une phrase latine et portant le nom "nomfichanal" dans une liste chaînée plus

facilement manipulable.
"firstlna" contiendra alors un pointeur vers le premier maillon de la chaîne.

14. **GESTCOMM**

Arguments : ligne
ldcour
texteb

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"ldcour" est un pointeur pointant vers la description générale d'un texte ou vers la description particulière d'une phrase de ce texte.
"texteb" est un booléen mentionnant s'il s'agit de la description d'un texte ("texteb" = 'true') ou d'une phrase ("texteb" = 'false').

Résultat : /

Postcondition : GESTCOMM affiche à partir de la ligne "ligne" soit "ldcour^.contenu.comment_txt", soit "ldcour^.contenu.comment_phrase" selon la valeur de "texteb". Et il propose de le garder, modifier ou annuler.

15. **TRT_VP**

Arguments : existfna
firstmcp
accesfamml
lnacour
firstlna
remonte

Préconditions : "existfna" est un booléen valant 'true' si le noeud d'analyse à traiter par TRT_VP existe déjà, sinon "existfna" = 'false'.
"firstmcp" est un pointeur pointant vers le premier maillon de la liste contenant la phrase à analyser.
"accesfamml" est un booléen valant 'true' si le fichier 'b:fmotlat.am' est déjà ouvert en lecture, sinon "accesfamml" vaut 'false'.

"lnacour" est un pointeur pointant vers le noeud d'analyse courant.
"firstlna" est un pointeur pointant vers le premier noeud d'analyse de la liste représentant l'Analyse de la phrase correspondante.
"remonte" est un booléen valant 'true' si l'on a effectué une remontée dans l'arbre d'analyse afin de traiter un noeud non encore considéré et devant l'être pour que l'Analyse soit complète.

Résultats : existfna
accesfamml
lnacour
firstlna
finanbool
finanal

Postconditions : TRTVP traite les noeuds d'analyse concernant le verbe principal : choix des candidats 'verbe principal', analyses morphologiques et de même pour les participes passés existant, dans le cas où le verbe principal traité est une forme de 'esse'.
Si "remonte" vaut 'true', selon le type de "lnacour", les traitements adéquats sont réalisés.
Selon la valeur de "existfna", certaines étapes de traitement peuvent être omises.
"lnacour" et éventuellement "firstlna" sont mis à jour selon l'avancement de l'analyse.
"existfna" et "accesfamml" sont également éventuellement modifiés selon l'analyse.
"finanbool" vaudra 'true' si le professeur décide de stopper momentanément son analyse ou si l'analyse de la phrase traitée est terminée, auquel cas, "finanal" vaudra aussi 'true'.

16. **TRT_GN**

Arguments : existfna
firstmcp
accesfamml
lnacour
casgn
remonte

Préconditions : "existfna" est un booléen valant 'true' si le noeud d'analyse à traiter par TRT_GN existe déjà, sinon "existfna" = 'false'.
"firstmcp" est un pointeur pointant vers le premier maillon de la liste contenant la phrase à analyser.
"accesfamml" est un booléen valant 'true' si le fichier 'b:fmotlat.am' est déjà ouvert en lecture, sinon "accesfamml" vaut 'false'.
"lnacour" est un pointeur pointant vers le noeud d'analyse courant.
"casgn" vaut 'nom', 'acc' ou 'abl' selon qu'il s'agit de traiter le groupe nominal sujet, cod ou complément.
"remonte" est un booléen valant 'true' si l'on a effectué une remontée dans l'arbre d'analyse afin de traiter un noeud non encore considéré et devant l'être pour que l'Analyse soit complète.

Résultats : existfna
accesfamml
lnacour
finanbool
noeudmem
trtsuivant
finanal

Postconditions : TRTGN traite les noeuds d'analyse concernant un groupe nominal : choix des candidats 'centraux', analyses morphologiques, choix de la découpe syntaxique et analyses morphologiques de tous les mots choisis.

Si "remonte" vaut 'true', selon le type de "lnacour", les traitements adéquats sont réalisés. Si "lnacour" n'est pas un noeud pouvant être traité par ce TRT_GN, "trtsuivant" sera mis à jour de manière à passer la main à un autre traitement susceptible de pouvoir considérer "lnacour", sinon "trtsuivant" vaudra la valeur par défaut, 'trtnul', de manière à indiquer un déroulement normal de l'analyse.
Selon la valeur de "existfna", certaines étapes de traitement peuvent être omises.
"lnacour" est mis à jour selon l'avancement de l'analyse.
"existfna" et "accesfamml" peuvent également être éventuellement modifiés selon l'analyse.
"finanbool" vaudra 'true' si le professeur décide de stopper momentanément son analyse ou si l'analyse de la phrase traitée est terminée, auquel cas, "finanal" vaudra aussi 'true'.
"noeudmem" vaudra 'nil' ou pointera vers un noeud d'analyse à raccrocher à l'arbre en cas de modification de celui-ci.

17. **TRT_TRAD**

Arguments : existfna
firstmcp
accesfamml
lnacour

Préconditions : "existfna" est un booléen valant 'true' si le noeud d'analyse à traiter par TRT_TRAD existe déjà, sinon "existfna" = 'false'.
"firstmcp" est un pointeur pointant vers le premier maillon de la liste contenant la phrase à analyser.
"accesfamml" est un booléen valant 'true' si le fichier 'b:fmotlat.am' est déjà ouvert en lecture, sinon "accesfamml" vaut 'false'.
"lnacour" est un pointeur pointant vers le noeud d'analyse courant.

Résultats : accesfamml
finanbool

Postconditions : TRT_TRAD traite les noeuds d'analyse concernant une traduction de la phrase concernée, selon les analyses morphologiques sélectionnées sur la branche d'analyse.
Selon la valeur de "existfna", certaines étapes de traitement peuvent être omises.
"accesfamml" peut éventuellement être modifié selon l'analyse.
"finanbool" vaudra 'true' si le professeur décide de stopper momentanément son analyse.

18. **MAJLNACOUR**

Arguments : firstmcp
lnacour

Préconditions : "firstmcp" est un pointeur pointant vers le premier maillon de la liste contenant la phrase à analyser.
"lnacour" est un pointeur pointant vers le noeud d'analyse courant.

Résultats : lnacour
existfna
remonte
fin

Postconditions : MAJLNACOUR parcourt l'arbre d'analyse en suivant l'ordre en profondeur d'abord et de gauche à droite, dans l'intention de trouver le prochain noeud d'analyse à traiter, à partir de "lnacour" (il faudra parfois recourir à une remontée dans l'arbre pour trouver ce noeud).
"lnacour" pointera alors vers ce noeud d'analyse à traiter ou vaudra 'nil' dans le cas où l'arbre d'analyse est complet, auquel cas "fin" vaudra alors 'true'.
Si "lnacour" est différent de 'nil', "existfna" vaudra 'true' si le noeud fils de "lnacour" existe, sinon il vaudra 'false'.
Et dans le cas où il aura fallu parcourir l'arbre d'analyse en le remontant, "remonte" vaudra 'true', sinon il vaudra 'false'.

19. **MAJLIBRE**

Arguments : firstmcp
noeud

Préconditions : "firstmcp" est un pointeur pointant vers le premier maillon de la liste contenant la phrase à analyser.
"noeud" est un noeud d'analyse de type 'n1'.

Résultat : /

Postconditions : MAJLIBRE met à jour la mention 'libre' dans le maillon appartenant à la liste accessible par "firstmcp" et correspondant au mot choisi pour une certaine fonction dans le noeud "noeud" ("noeud.mot1").

20. **AFFCOMM**

Arguments : ligne
comment
existfna

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"comment" est un string[80].
"existfna" est un booléen mentionnant si le noeud d'analyse dont le commentaire spécifique "comment" est considéré, existe déjà ou pas.

Résultat : comment

Postconditions : si "existfna" = 'true', AFFCOMM affiche à partir de la ligne "ligne", "comment" et offre la possibilité de le modifier,
si "existfna" = 'false', AFFCOMM demande l'introduction de "comment" à partir de la ligne "ligne".

21. **AFFACCCOMM**

Arguments : ligne
accept
existfna

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"accept" est un booléen.

"existfna" est un booléen mentionnant si le noeud d'analyse dont la mention d'acceptation du commentaire "accept" est considérée, existe déjà ou pas.

Résultat : accept

Postconditions : si "existfna" = 'true', AFFACCCOMM affiche à partir de la ligne "ligne", la mention d'acceptation du commentaire ("accept") et offre la possibilité de la modifier,
si "existfna" = 'false', AFFACCCOMM demande l'introduction de la mention d'acceptation "accept" à partir de la ligne "ligne".

22. **AFFINDBRK**

Arguments : ligne
lnoeudaa
existfna
chemin

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"lnoeudaa" est un pointeur pointant vers le noeud d'analyse considéré.
"existfna" est un booléen mentionnant si le noeud d'analyse dont la mention 'indicatif ou break' est considérée, existe déjà ou pas.
"chemin" est un booléen mentionnant si les noeuds appartenant au chemin aboutissant à ce noeud correspondent à une analyse de la phrase correcte ou pas.

Résultats : lnoeudaa
chemin

Postconditions : si "existfna" = 'true', AFFINDBRK affiche à partir de la ligne "ligne", la mention du type d'étape d'analyse ("lnoeudaa^.noeud.ind_brk") et offre la possibilité de la modifier;
si "existfna" = 'false', AFFINDBRK demande l'introduction de la mention du type d'étape d'analyse "lnoeudaa^.noeud.ind_brk" à partir de la ligne "ligne".

"chemin" sera éventuellement modifié. Et la mention indiquant l'appartenance d'un noeud à une analyse correcte de la phrase sera également éventuellement mise à jour pour chaque noeud 'ancêtre' du noeud considéré.

23. **FIND_INDEX**

Argument : mot
Préconditions : "mot" est un string[20] représentant une forme latine existant dans le fichier 'b:fmotlat.am'.
Résultat : where
Postconditions : "where" prend la valeur de la position de la première analyse morphologique possible de la forme latine mot dans le fichier 'b:fmotlat.am'.

24. **TRSE**

Argument : noeud
Précondition : "noeud" est noeud d'analyse de type 'n2'.
Résultat : analyse
Postcondition : "analyse" est un string[10] créé à partir des informations décrivant une analyse morphologique dans le noeud "noeud". "analyse" sera conforme au code élaboré par le LASLA de Liège.

25. **REMPLI_AM_V**

Arguments : ligne
noeud
Préconditions : "ligne" contient des informations décrivant une analyse morphologique verbale.
"noeud" est un noeud d'analyse de type 'n2'.
Résultat : noeud
Postcondition : REMPLI_AM_V remplit la partie décrivant une analyse morphologique dans le noeud "noeud", à partir des informations enregistrées dans "ligne".

26. **AFFICH_AM_V**

Arguments : ligne
noeud

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"noeud" est un noeud d'analyse de type 'n2', contenant la description d'une analyse morphologique verbale.

Résultat : /

Postcondition : AFFICH_AM_V affiche à partir de la ligne "ligne" jusqu'à la ligne "ligne + 3", les informations morphologiques contenues dans "noeud".

27. **REMPLI_AM_SUBST**

Arguments : ligne
noeud

Préconditions : "ligne" contient des informations décrivant une analyse morphologique d'un substantif.
"noeud" est un noeud d'analyse de type 'n2'.

Résultat : noeud

Postcondition : REMPLI_AM_SUBST remplit la partie décrivant une analyse morphologique dans le noeud "noeud", à partir des informations enregistrées dans "ligne".

28. **AFFICH_AM_SUBST**

Arguments : ligne
noeud

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"noeud" est un noeud d'analyse de type 'n2', contenant la description d'une analyse morphologique d'un substantif.

Résultat : /

Postcondition : AFFICH_AM_SUBST affiche à partir de la ligne "ligne" jusqu'à la ligne "ligne + 3", les informations morphologiques contenues dans "noeud".

29. **REMPLI_AM_ADJ**

Arguments : ligne
noeud

Préconditions : "ligne" contient des informations décrivant une analyse morphologique d'un adjectif.
"noeud" est un noeud d'analyse de type 'n2'.

Résultat : noeud

Postcondition : REMPLI_AM_ADJ remplit la partie décrivant une analyse morphologique dans le noeud "noeud", à partir des informations enregistrées dans "ligne".

30. **AFFICH_AM_ADJ**

Arguments : ligne
noeud

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"noeud" est un noeud d'analyse de type 'n2', contenant la description d'une analyse morphologique d'un adjectif.

Résultat : /

Postcondition : AFFICH_AM_ADJ affiche à partir de la ligne "ligne" jusqu'à la ligne "ligne + 3", les informations morphologiques contenues dans "noeud".

31. **REMPLI_AM_ADJPRON**

Arguments : ligne
noeud

Préconditions : "ligne" contient des informations décrivant une analyse morphologique d'un adjectif pronom.
"noeud" est un noeud d'analyse de type 'n2'.

Résultat : noeud

Postcondition : REMPLI_AM_ADJPRON remplit la partie décrivant une analyse morphologique dans le noeud "noeud", à partir des informations enregistrées dans "ligne"

32. **AFFICH_AM_ADJPRON**

Arguments : ligne
noeud

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"noeud" est un noeud d'analyse de type 'n2', contenant la description d'une analyse morphologique d'un adjectif pronom.

Résultat : /

Postcondition : AFFICH_AM_ADJPRON affiche à partir de la ligne "ligne" jusqu'à la ligne "ligne + 3", les informations morphologiques contenues dans "noeud".

33. **REMPLI_AM_PREP**

Arguments : ligne
noeud

Préconditions : "ligne" contient des informations décrivant une analyse morphologique d'une préposition.
"noeud" est un noeud d'analyse de type 'n2'.

Résultat : noeud

Postcondition : REMPLI_AM_PREP remplit la partie décrivant une analyse morphologique dans le noeud "noeud", à partir des informations enregistrées dans "ligne".

34. **AFFICH_AM_PREP**

Arguments : ligne
noeud

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"noeud" est un noeud d'analyse de type 'n2', contenant la description d'une analyse morphologique d'une préposition.

Résultat : /

Postcondition : AFFICH_AM_PREP affiche à partir de la ligne "ligne" jusqu'à la ligne "ligne + 1", les informations morphologiques contenues dans "noeud".

35. **REMPLI_AM_CONJ**

Arguments : ligne
noeud

Préconditions : "ligne" contient des informations décrivant une analyse morphologique d'une conjonction.
"noeud" est un noeud d'analyse de type 'n2'.

Résultat : noeud

Postcondition : REMPLI_AM_CONJ remplit la partie décrivant une analyse morphologique dans le noeud "noeud", à partir des informations enregistrées dans "ligne".

36. **AFFICH_AM_CONJ**

Arguments : ligne
noeud

Préconditions : "ligne" est un entier appartenant à l'intervalle [1..25].
"noeud" est un noeud d'analyse de type 'n2', contenant la description d'une analyse morphologique d'une conjonction.

Résultat : /

Postcondition : AFFICH_AM_CONJ affiche à partir de la ligne "ligne" jusqu'à la ligne "ligne + 3", les informations morphologiques contenues dans "noeud".

37. **REMPLI_N3_LN3**

Arguments : firstln3
firstmcp

Préconditions : "firstln3" est un pointeur pointant vers le premier maillon d'une chaîne décrivant un ensemble de mots. Cet ensemble de mots est inclu dans celui décrit par la liste accessible par "firstmcp".
"firstmcp" est un pointeur pointant vers le premier maillon d'une chaîne représentant une phrase à analyser.

Résultat : ensmots3

Postcondition : à partir des deux listes accessibles respectivement par "firstln3" et "firstmcp", REMPLI_N3_LN3 remplit le tableau de booléens "ensmots3" de manière à renseigner quels sont les mots de la liste accessible par "firstmcp" mentionnés dans la liste accessible par "firstln3" (= booléen = true).

38. **TRSF_N3_LN3**

Arguments : firstmcp
ensmots3

Préconditions : "firstmcp" est un pointeur pointant vers le premier maillon d'une chaîne représentant une phrase à analyser.
"ensmots3" est un tableau de booléens associé à la liste de mots accessible par "firstmcp"; chaque booléen vaut 'true' si le mot correspondant est sélectionné, sinon il vaut 'false'.

Résultat : firstln3

Postcondition : à partir du tableau "ensmots3" et de la liste accessible par "firstmcp", TRSF_N3_LN3 crée la liste de mots accessible par "firstln3".

39. **AFFICHN3**

Arguments : noeud
firstmcp
miny
maxy

Préconditions : "noeud" est un noeud d'analyse de type 'n3'.
"firstmcp" est un pointeur pointant vers le premier maillon d'une chaîne représentant une phrase à analyser.
"miny" et "maxy" sont des entiers appartenant à l'intervalle [1..25].

Résultat : /

Postcondition : AFFICHN3 affiche entre les lignes "miny" et "maxy", les mots correspondants aux informations enregistrées dans "noeud.ensmots3", et cela, à partir de la liste de mots accessible par "firstmcp".

40. **AFFICHLN3**

Arguments : firstln3
miny
maxy

Préconditions : "firstln3" est un pointeur pointant vers le premier maillon d'une liste de mots.
"miny" et "maxy" sont des entiers appartenant à l'intervalle [1..25].

Résultat : /

Postcondition : AFFICHLN3 affiche entre les lignes "miny" et "maxy", les mots sélectionnés dans la liste accessible par "firstln3".

Remarque :

Toute personne intéressée par le code de chaque procédure, écrit en Turbo Pascal 4.0, aura recours aux annexes 9, 10, 11, 12, 13, 14 et 15.

7.3- Exploitation des leçons préparées

7.3.1- Fichier contenant l'analyse d'une phrase latine élaborée par un élève : description

Cette partie du didacticiel est destinée aux élèves et leur permet d'analyser une ou plusieurs phrases latines de leur choix. Ce choix est cependant limité aux phrases latines mises à leur disposition. Et, il est également nécessaire que le fichier contenant l'analyse complète, préparée par l'enseignant, existe.

Comme annoncé auparavant, c'est à partir du fichier d'analyse 'b:nomfichierlasla.numerophrase', créé lors de la phase précédente, que le dialogue avec l'élève a lieu. Ce dialogue a pour but d'aider l'élève à élaborer une analyse de la phrase traitée.
Tout comme dans la phase de préparation, l'analyse de l'élève est aussi représentée sous forme d'un arbre composé de noeuds, équivalents chacun à une étape d'analyse. Et ces informations sont également manipulées par l'intermédiaire d'une liste chaînée lors du traitement. Elles sont, ensuite, mémorisées dans un fichier d'analyse (**'b:nomfichierlasla.numerophrase'EL''**). Les structures de ce fichier et de la liste chaînée sont quasi-identiques à celles du fichier d'analyse et de la liste chaînée correspondante, gérés lors de la phase de préparation (cfr. infra 7.2.1- Fichier d'analyse d'une phrase latine : description). La différence réside dans le fait que chaque record, étape d'analyse d'un élève, ne contient que les informations décrivant le type du noeud ('n1', 'n2', 'n3' ou 'n4') et les données y associées (cfr. p.70). Les autres renseignements sont seulement précisés dans la phase d'élaboration de leçons, pour être utilisés ultérieurement dans le dialogue avec l'élève.

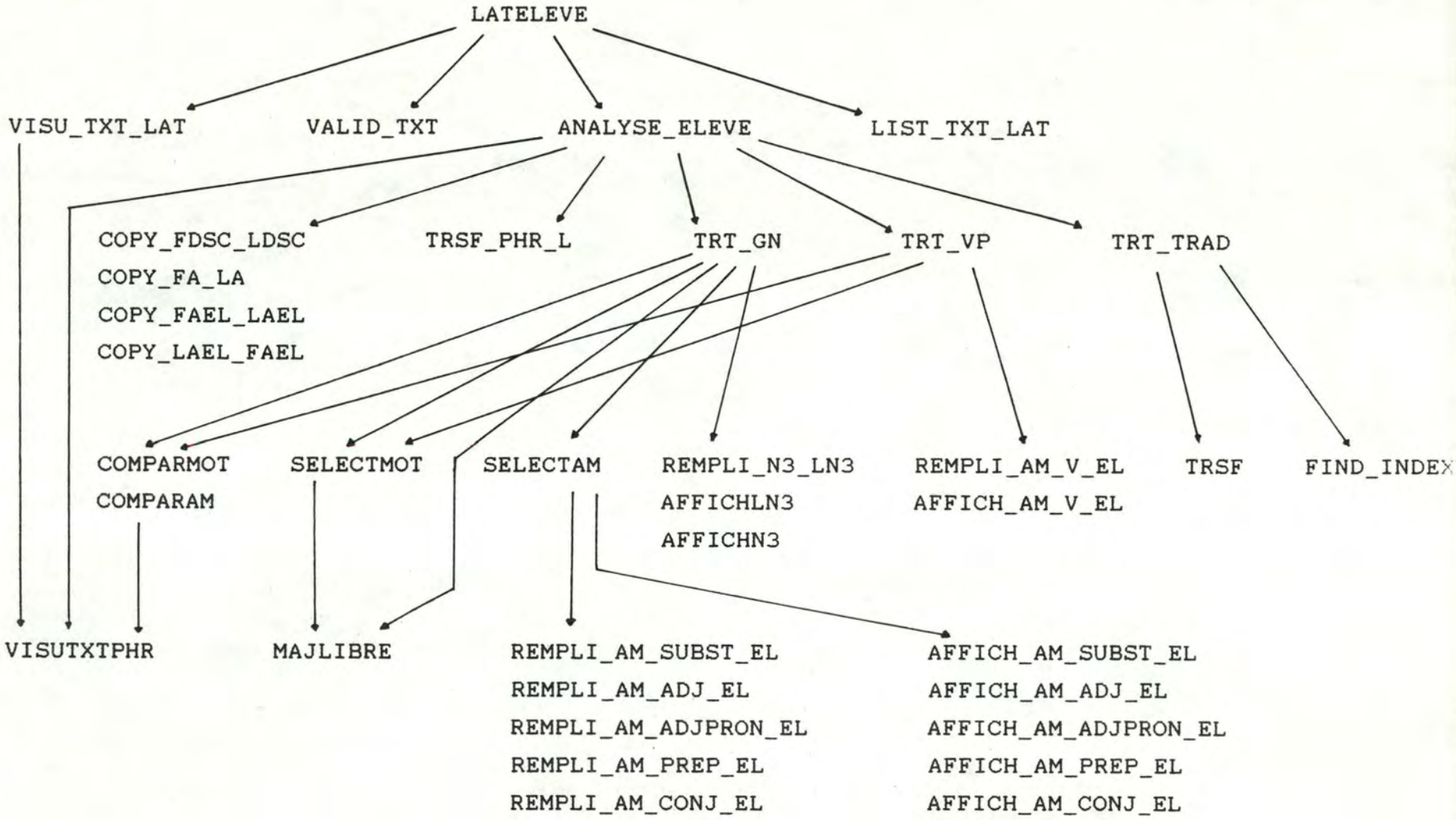
Avant de poursuivre, rappelons que nous avons limité à un, le nombre de traductions françaises associées à chaque analyse morphologique de chaque forme latine appartenant au fichier 'b:fmotlatin.am'. Il n'est, alors, pas possible de proposer à l'élève, comme nous avions initialement penser le faire, le choix entre plusieurs traductions valables pour un même mot, selon une certaine analyse morphologique sélectionnée. Ceci est en effet irréalisable puisqu'il n'existe qu'une telle traduction disponible dans le fichier 'b:fmotlat.am' connu du système.
L'alternative serait de demander à l'élève d'introduire sa traduction pour chaque mot. Cette traduction serait ensuite comparée avec celle prévue dans 'b:fmotlat.am'. Mais cette solution est assez "primaire". Pour s'en convaincre, il suffit de savoir que, par exemple, les synonymes ou les mots non exactement orthographiés comme il l'est prévu (ex. majuscules ou blancs absents ou superflus) sont refusés à l'élève alors qu'ils peuvent être sémantiquement corrects !
Voilà pourquoi nous avons adopté une autre solution concernant la gestion de la traduction des phrases latines analysées. Le système sélectionne dans 'b:fmotlat.am' une traduction pour chaque mot de la phrase. Cette sélection est établie à partir de l'analyse morphologique choisie par l'élève, pour chaque mot de la phrase, lors des différentes étapes d'analyse. Dès lors, apparaît à l'écran la traduction de la phrase sous forme de l'apposition de ces traductions. L'élève peut, s'il le désire, connaître les associations "forme latine-traduction française" présentées par le système. Cette information lexicale renseigne l'élève quant à l'exactitude de son analyse de la phrase, puisque la traduction affichée est sensée ou non.

Ceci étant précisé, nous pouvons à présent décrire les procédures intervenant dans cette deuxième phase du didacticiel.

7.3.2- Spécifications des procédures

Détaillons, donc, les procédures utilisées par le programme **LATELEVE**. Celui-ci offre à l'élève, dans son menu, les possibilités de visualiser un texte latin, de voir la liste des fichiers contenant chacun un texte latin et d'analyser une ou plusieurs phrases latines.

Vous trouverez à la page suivante le graphe structurant ces procédures d'après la relation "appelle" définie précédemment.


LATELEVE
VISU_TXT_LAT
VALID_TXT
ANALYSE_ELEVE
LIST_TXT_LAT
COPY_FDSC_LDSC
COPY_FA_LA
COPY_FAEL_LAEL
COPY_LAEL_FAEL
TRSF_PHR_L
TRT_GN
TRT_VP
TRT_TRAD
COMPARMOT
COMPARAM
SELECTMOT
SELECTAM
REMPLI_N3_LN3
AFFICHLN3
AFFICHN3
REMPLI_AM_V_EL
AFFICH_AM_V_EL
TRSF
FIND_INDEX
VISUTXTPHR
MAJLIBRE
REMPLI_AM_SUBST_EL
REMPLI_AM_ADJ_EL
REMPLI_AM_ADJPRON_EL
REMPLI_AM_PREP_EL
REMPLI_AM_CONJ_EL
AFFICH_AM_SUBST_EL
AFFICH_AM_ADJ_EL
AFFICH_AM_ADJPRON_EL
AFFICH_AM_PREP_EL
AFFICH_AM_CONJ_EL

1. "VISUALISATION TEXTE LATIN" : **VISU_TXT_LAT**

cfr. 2. "VISUALISATION TEXTE LATIN" : VISU_TXT_LAT (p.72)

2. **VALID_TXT**

cfr. 7. VALID_TXT (p.77)

3. "ANALYSE TEXTE LATIN PAR UN ELEVE" : **ANALYSE_ELEVE**

Argument : namefct

Préconditions : "namefct" est un string[14] représentant le nom d'un fichier 'b:*.fct' existant et contenant un texte latin.
Pour un bon déroulement de la procédure, sont nécessaires le fichier 'b:*.dsc' associé, ainsi que les fichiers contenant les analyses des phrases de ce texte, préparées par l'enseignant. Les fichiers contenant les analyses des phrases du texte, déjà (en partie) élaborées par l'élève, ainsi que les fichiers généraux 'b:fmotlat.am' et 'b:index.am' doivent être aussi être présents sur la disquette en drive B.

Résultat : ANALYSE_ELEVE offre à l'élève la possibilité d'établir ou mettre à jour l'analyse d'une ou plusieurs phrases du texte contenu dans le fichier portant le nom "namefct". Ceci, sachant que seules les phrases dont les analyses ont été préparées "complètement" par l'enseignant, pourront faire l'objet de ces exercices auprès de l'élève.

Postcondition : les fichiers contenant les analyses des phrases élaborées par l'élève seront mis à jour ou créés.

4. "LISTE TEXTES LATINS : **LIST_TXT_LAT**

cfr. 1. "LISTE TEXTES LATINS" : LIST_TXT_LAT (p.72)

5. **COPY_FDSC_LDSC**

cfr. 10. COPY_FDSC_LDSC (p.78)

6. **COPY_FA_LA**

cfr. 13. COPY_FA_LA (p.79)

7. **COPY_FAEL_LAEL**

Argument : nomfichanalel

Préconditions : "nomfichanael" est un string[14] représentant le fichier existant contenant l'analyse d'une phrase latine, élaborée par l'élève.

Résultat : firstlnael

Postconditions : COPY_FAEL_LAEL copie le contenu du fichier décrivant l'analyse d'une phrase latine élaborée par un élève et portant le nom "nomfichanalel", dans une liste chaînée plus facilement manipulable.
"firstlnael" contiendra alors un pointeur pointant vers le premier maillon de cette chaîne.

8. **COPY_LAEL_FAEL**

Arguments : nomfichanalel
firstlnael

Préconditions : "nomfichanael" est un string[14].
"firstlnael" est un pointeur pointant vers le premier maillon de la liste chaînée contenant les informations décrivant l'analyse (complète ou non) d'une phrase latine élaborée par un élève.

Résultat : nomfichanael

Postconditions : COPY_LAEL_FAEL copie dans le fichier décrivant l'analyse d'une phrase latine élaborée par un élève et portant le nom "nomfichanalel", les informations y associées, contenues dans la liste chaînée dont le premier maillon est accessible par "firstlnael".

9. **TRSF_PHR_L**

cfr. 9. TRSF_PHR_L (p.78)

10. **TRT_VP**

Arguments : existfnael
firstmcp
lnacour
firstlna
lnacourel
firstlnael
remonte
namefct
titretxt
firstld
ldcour

Préconditions : "existfnael" est un booléen valant 'true' si le noeud d'analyse de l'élève à traiter par TRT_VP existe déjà, sinon "existfnael" vaut 'false'.
"firstmcp" est un pointeur pointant vers le premier maillon de la chaîne contenant la phrase à analyser.
"lnacour" est un pointeur pointant vers le noeud d'analyse courant dans la liste chaînée contenant l'analyse de la phrase préparée par l'enseignant.
"firstlna" est un pointeur pointant vers le premier maillon de la liste chaînée contenant l'analyse de la phrase préparée par l'enseignant.
"lnacourel" est un pointeur pointant vers le noeud d'analyse courant dans la liste chaînée contenant l'analyse de la phrase élaborée par l'élève.
"firstlnael" est un pointeur pointant vers le premier maillon de la liste chaînée contenant l'analyse de la phrase élaborée par l'élève.
"remonte" est un booléen valant 'true' si l'élève a effectué une remontée dans son arbre d'analyse.
"namefct" est un string[14] équivalent au nom du fichier contenant le texte latin d'où provient la phrase à analyser.
"titretxt" est un string[60] contenant le titre donné par l'enseignant au texte contenu dans le fichier portant le nom "namefct".

"firstld" est un pointeur pointant vers le premier maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce premier maillon décrit d'une manière générale le texte et contient notamment le commentaire général associé à ce texte par l'enseignant.

"ldcour" est un pointeur pointant vers un maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce maillon décrit plus particulièrement la phrase traitée et contient notamment le commentaire associé à cette phrase par l'enseignant.

Résultats : existfnael
lnacour
lnacourel
firstlnael
finanbool

Postconditions : TRT_VP gère les étapes d'analyse de l'élève concernant le verbe principal : choix d'un verbe principal, analyse morphologique et de même pour un participe passé dans le cas où le verbe choisi est une forme du verbe 'esse'. Cette gestion et le dialogue avec l'élève sont facilités grâce à l'Analyse associée, préparée par l'enseignant.

"lnacourel" et "lnacour" sont mis à jour parallèlement selon l'avancement de l'analyse.

"firstlnael" est également éventuellement modifié dans le cas où "existfnael" vaut 'false' à l'origine.

"existfnael" est mis à jour continuellement selon l'existence de l'étape d'analyse de l'élève à traiter.

Si "remonte" vaut 'false', le choix des traitements des étapes d'analyse est modifié.

"finanbool" vaudra 'true' si l'élève décide de stopper son analyse en cours, sinon il vaudra 'false'.

"namefct" et "titretxt" sont utilisés dans le cas où l'élève désire (re)voir le texte d'où provient

la phrase analysée, au cours de l'analyse.
"firstld" et "ldcour" sont, quant à eux, utilisés dans le cas où l'élève désire (re)voir le commentaire général du texte ou le commentaire spécifique de la phrase analysée, en cours d'analyse.

11. **TRT_GN**

Arguments : existfnael
firstmcp
lnacour
lnacourel
casgn
remonte
namefct
titretxt
firstld
ldcour

Préconditions : "existfnael" est un booléen valant 'true' si le noeud d'analyse de l'élève à traiter par TRT_GN existe déjà, sinon "existfnael" vaut 'false'.
"firstmcp" est un pointeur pointant vers le premier maillon de la chaîne contenant la phrase à analyser.
"lnacour" est un pointeur pointant vers le noeud d'analyse courant dans la liste chaînée contenant l'Analyse de la phrase préparée par l'enseignant.
"lnacourel" est un pointeur pointant vers le noeud d'analyse courant dans la liste chaînée contenant l'analyse de la phrase élaborée par l'élève.
"casgn" vaut 'nom', 'acc' ou 'abl' selon qu'il s'agit de traiter le groupe nominal sujet, cod ou complément.
"remonte" est un booléen valant 'true' si l'élève a effectué une remontée dans son arbre d'analyse.
"namefct" est un string[14] équivalent au nom du fichier contenant le texte latin d'où provient la phrase à analyser.

"titretxt" est un string[60] contenant le titre donné par l'enseignant au texte contenu dans le fichier portant le nom "namefct".

"firstld" est un pointeur pointant vers le premier maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce premier maillon décrit d'une manière générale le texte et contient notamment le commentaire général associé à ce texte par l'enseignant.

"ldcour" est un pointeur pointant vers un maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce maillon décrit plus particulièrement la phrase traitée et contient notamment le commentaire associé à cette phrase par l'enseignant.

Résultats : existfnael
lnacour
lnacourel
finanbool
trtsuivant

Postconditions : TRT_GN gère les étapes d'analyse de l'élève concernant un groupe nominal : choix du centre de ce groupe, analyse morphologique, découpe syntaxique et analyse morphologique de chaque mot choisi appartenant à ce groupe. De même que pour TRT_VP, cette gestion et le dialogue avec l'élève sont facilités grâce à l'Analyse associée, préparée par l'enseignant.

"lnacourel" et "lnacour" sont mis à jour parallèlement selon l'avancement de l'analyse.

"existfnael" est mis à jour continuellement selon l'existence de l'étape d'analyse de l'élève à traiter.

Si "remonte" vaut 'false', le choix des traitements des étapes d'analyse est modifié.

"finanbool" vaudra 'true' si l'élève décide de stopper son analyse en cours, sinon il vaudra 'false'.

"trtsuivant" sera mis à jour dans le cas où l'élève remonte dans son arbre d'analyse et que l'étape d'analyse courante ne peut pas être traitée par la procédure TRT_GN en cours :
si "casgn" vaut 'nom'/'acc'/'abl',
"trtsuivant" vaudra 'trtv'/'trtgns'/'trtgncod'.
Si l'élève ne remonte pas dans son arbre d'analyse, "trtsuivant" vaudra 'trtnul', de manière à indiquer que le déroulement normal de l'analyse peut s'effectuer.
"namefct" et "titretxt" sont utilisés dans le cas où l'élève désire (re)voir le texte d'où provient la phrase analysée, au cours de l'analyse.
"firstld" et "ldcour" sont, quant à eux, utilisés dans le cas où l'élève désire (re)voir le commentaire général du texte ou le commentaire spécifique de la phrase analysée, en cours d'analyse.

12. **TRT_TRAD**

Arguments : firstmcp
accesfamml
lnacour

Préconditions : "firstmcp" est un pointeur pointant vers le premier maillon de la chaîne contenant la phrase à analyser.
"accesfamml" est un booléen valant 'true' si le fichier 'b:fmotlat.am' a déjà été ouvert en lecture, sinon il vaut 'false'.
"lnacour" est un pointeur pointant vers le noeud d'analyse courant dans la liste chaînée contenant l'Analyse de la phrase préparée par l'enseignant.

Résultats : accesfamml
finanbool
lnacour

Postconditions : TRT_TRAD gère l'étape d'analyse correspondant à la traduction de la phrase selon l'analyse élaborée par l'élève. Cette gestion et le dialogue avec l'élève sont réalisés à partir

des informations contenues dans l'arbre d'analyse de la phrase préparé par l'enseignant "accesfamml" vaudra 'true'.
"finanbool" vaudra 'true' si la traduction correspond à l'étape d'analyse clôturant une analyse correcte de la phrase élaborée par l'élève ou si l'analyse n'étant pas correcte, l'élève décide de stopper son exercice,
sinon, "finanbool" vaudra 'false'.
Si cette étape d'analyse n'est pas la confirmation d'une analyse correcte, "lnacour" est mis à jour de manière à être conforme pour aborder une remontée dans l'arbre d'analyse de l'élève, en vue de correction afin de trouver une analyse exacte de la phrase.

13. **COMPARMOT**

Arguments : firstlna
lnacourel
firstmcp
firstlnael
namefct
titretxt
firstld
ldcour
comment

Préconditions : "firstlna" est un pointeur pointant vers un noeud d'analyse dans la liste chaînée contenant l'Analyse de la phrase préparée par l'enseignant. Ce noeud d'analyse représente le premier noeud d'analyse de type 'n1' susceptible de concorder avec l'étape d'analyse introduite par l'élève (= "lnacourel").
"lnacourel" est un pointeur pointant vers le noeud d'analyse courant de type 'n1' représentant l'étape d'analyse venant d'être traitée par l'élève.
"firstmcp" est un pointeur pointant vers le premier maillon de la chaîne contenant la phrase à analyser.

"firstlnael" est un pointeur pointant vers un noeud d'analyse de type 'n1' appartenant à la liste chaînée représentant l'analyse de la phrase élaborée par l'élève.
"namefct" est un string[14] équivalent au nom du fichier contenant le texte latin d'où provient la phrase à analyser.
"titretxt" est un string[60] contenant le titre donné par l'enseignant au texte contenu dans le fichier portant le nom "namefct".
"firstld" est un pointeur pointant vers le premier maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce premier maillon décrit d'une manière générale le texte et contient notamment le commentaire général associé à ce texte par l'enseignant.
"ldcour" est un pointeur pointant vers un maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce maillon décrit plus particulièrement la phrase traitée et contient notamment le commentaire associé à cette phrase par l'enseignant.
"comment" est un string[80].

Résultats : lnacour
lnacourel
finanbool
gotofin
existfnael
gotodebut

Postconditions : COMPARMOT vérifie si le mot sélectionné lors d'une étape d'analyse par l'élève ("lnacourel^.noeudel.motiel") concorde avec un des mots prévus dans l'Analyse préparée et enregistrés dans les noeuds d'analyse "firstlna^.noeud" et frères.
S'il n'existe pas de concordance, le commentaire "comment" s'affiche, "gotodebut" et "existfnael" prennent la valeur 'true' et "lnacourel" devient "firstlnael", dans le but

de pouvoir ensuite sélectionner un autre mot susceptible d'être parmi ceux prévus par l'Analyse préparée.
S'il y a concordance et que cette étape d'analyse correspond à un noeud 'break', les mêmes conséquences que ci-dessus se produisent quant à "gotodebut", "existfnael" et "lnacourel";
il se passera encore les mêmes choses si le noeud est 'indicatif' mais que l'élève décide de recommencer cette étape d'analyse;
tandis que s'il s'agit d'un noeud 'indicatif', "namefct", "firstld" et "ldcour" pourront être utilisés dans le cas où l'élève désire (re)voir le texte d'où provient la phrase à analyser, son commentaire ou le commentaire de la phrase.
"lnacour" pointera soit sur le noeud d'analyse concordant à celui de l'élève, soit sur 'nil' dans le cas où aucune concordance n'existe.
"finanbool" vaudra 'true' dans le cas où l'élève décide de stopper son analyse, auquel cas, "gotofin" vaudra aussi 'true', sinon ils vaudront tous deux 'false'.

14. **COMPARAM**

Arguments : lnacour
noeudfilscourel
categorie
namefct
titretxt
firstld
ldcour

Préconditions : "lnacour" est un pointeur pointant vers un noeud d'analyse de type 'n1' dans la liste chaînée contenant l'analyse de la phrase préparée par l'enseignant.
"noeudfilscourel" est un pointeur pointant vers le noeud d'analyse courant de type 'n2' représentant l'étape d'analyse venant d'être

traitée par l'élève.

"categorie" vaudra 'v', 'subst', 'adj', 'adjpron', 'prep' ou 'conj' selon que l'analyse morphologique décrite dans le noeud "lnacourel^.noeudel" qualifie un verbe, un substantif, un adjectif, un adjectif pronom, une préposition ou une conjonction.

"namefct" est un string[14] équivalent au nom du fichier contenant le texte latin d'où provient la phrase à analyser.

"titretxt" est un string[60] contenant le titre donné par l'enseignant au texte contenu dans le fichier portant le nom "namefct".

"firstld" est un pointeur pointant vers le premier maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce premier maillon décrit d'une manière générale le texte et contient notamment le commentaire général associé à ce texte par l'enseignant.

"ldcour" est un pointeur pointant vers un maillon de la liste chaînée contenant des informations décrivant le texte contenu dans le fichier portant le nom "namefct". Ce maillon décrit plus particulièrement la phrase traitée et contient notamment le commentaire associé à cette phrase par l'enseignant.

Résultats : noeudfilscour
lnacour
noeudfilscourel
lnacourel
finanbool
gotofin
gotoanal
existfnael

Postconditions : COMPARAM vérifie si l'analyse morphologique sélectionnée lors d'une étape d'analyse par l'élève ("noeudfilscourel^.noeudel") concorde avec une de celles prévues dans l'Analyse préparée et enregistrées dans les noeuds d'analyse "lnacour^.noeud" et frères.

S'il n'existe pas de concordance ou si le noeud d'analyse concordant est 'break', "gotoanal" et "existfnael" prennent la valeur 'true' et "noeudfilscourel" devient "lnacourel^.noeudfilsel", dans le but de pouvoir ensuite sélectionner une autre analyse morphologique susceptible d'être parmi celles prévues par l'Analyse préparée. (dans le cas où il y a eu concordance 'break', "lnacourel" et "lnacour" sont également mis à jour).
Il se passera encore les mêmes choses si le noeud est 'indicatif' mais que l'élève décide de recommencer cette étape d'analyse;
tandis que s'il s'agit d'un noeud 'indicatif', "namefct", "firstld" et "ldcour" pourront être utilisés dans le cas où l'élève désire (re)voir le texte d'où provient la phrase à analyser, son commentaire ou le commentaire de la phrase.
"noeudfilscour" pointera soit sur le noeud d'analyse concordant à celui de l'élève, soit sur 'nil' dans le cas où aucune concordance n'existe.
En cas de concordance, "lnacourel" et "lnacour" pointeront également sur l'analyse morphologique en question; ils vaudront donc respectivement "noeudfilscourel" et "noeudfilscour".
"finanbool" vaudra 'true' dans le cas où l'élève décide de stopper son analyse, auquel cas, "gotofin" vaudra aussi 'true', sinon ils vaudront tous deux 'false'.

15. **SELECTMOT**

Arguments : firstmcp
existfnael
fonction
entete
lnacourel
nodepereel

Préconditions : "firstmcp" est un pointeur pointant vers le premier maillon de la liste chaînée contenant la phrase à analyser.
"existfnael" est un booléen valant 'true' si le noeud d'analyse correspondant à l'étape d'analyse élaborée par l'élève et devant être traitée par SELECTMOT existe déjà, sinon il vaut 'false'.
"fonction" vaut 'verbe', 'sujet', 'cod' ou 'cplt' selon que le mot à sélectionner dans cette étape d'analyse dépend du Verbe Principal, du GNsujet, du GNcod ou du GNcomplément.
"entete" est un string[30].
Si "existfnael" vaut 'true', "lnacourel" pointe vers le noeud d'analyse courant de type 'n1' représentant l'étape d'analyse existante élaborée par l'élève, à traiter dans SELECTMOT.
"noeudpereel" est un pointeur pointant vers le noeud d'analyse représentant l'étape d'analyse précédent celle-ci et élaborée par l'élève.

Résultats : existfnael
lnacourel
firstlnael
remonte

Postconditions : SELECTMOT gère l'étape d'analyse équivalent au choix d'un mot de la phrase, répondant à une certaine fonction ("fonction").
Selon la valeur de "existfnael", divers traitements sont effectués.
"existfnael" est continuellement mis à jour en fonction de l'existence du noeud d'analyse de l'élève à traiter.
"entete" intervient dans le dialogue avec l'élève.
Si l'élève sélectionne un mot, "lnacourel" équivaudra au noeud contenant les informations associées à cette étape d'analyse correspondant à ce choix.
Dans le cas où la valeur initiale de "existfnael" vaut 'false' et que l'élève

effectue la sélection demandée, "firstlnael" sera mis à jour et égalera le noeud d'analyse créé.
Si l'élève décide de ne pas réaliser cette étape d'analyse et de remonter dans l'arbre d'analyse, "remonte" prendra la valeur 'true', sinon il vaudra 'false'.

16. **SELECTAM**

Arguments : entete
motanalyse
existfnael
noeudfilscourel
lnacourel

Préconditions : "entete" est un string[30].
"motanalyse" est un string[20] représentant le mot de la phrase dont l'analyse morphologique est considérée dans cette étape d'analyse.
"existfnael" est un booléen valant 'true' si le noeud d'analyse correspondant à l'étape d'analyse élaborée par l'élève et devant être traitée par SELECTAM existe déjà, sinon il vaut 'false'.
Si "existfnael" vaut 'true', "noeudfilscourel" pointe vers le noeud d'analyse courant de type 'n2' représentant l'étape d'analyse existante élaborée par l'élève, à traiter dans SELECTAM.
"lnacourel" est un pointeur pointant vers le noeud d'analyse de type 'n1' représentant l'étape d'analyse précédent celle-ci et élaborée par l'élève.

Résultats : existfnael
noeudfilscourel
remonte

Postconditions : SELECTAM gère l'étape d'analyse équivalent au choix d'une analyse morphologique du mot de la phrase "motanalyse".
Selon la valeur de "existfnael", divers traitements sont effectués.

"existfnael" est continuellement mis à jour en fonction de l'existence du noeud d'analyse de l'élève à traiter.
"entete" intervient dans le dialogue avec l'élève.
Si l'élève sélectionne une analyse morphologique, "noeudfilscourel" équivaudra au noeud contenant les informations associées à cette étape d'analyse correspondant à ce choix.
Dans le cas où la valeur initiale de "existfnael" vaut 'false' et que l'élève effectue la sélection demandée, "lnacourel^.noeudfilsel" sera mis à jour et égalera le noeud d'analyse créé.
Si l'élève décide de ne pas réaliser cette étape d'analyse et de remonter dans l'arbre d'analyse, "remonte" prendra la valeur 'true', sinon il vaudra 'false'.

17. **REMPLI_N3_LN3**

cfr. 37. REMPLI_N3_LN3 (p.91)

18. **AFFICHLN3**

cfr. 40. AFFICHLN3 (p.93)

19. **AFFICHN3**

Arguments : noeudel
firstmcp
miny
maxy

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n3'.
"firstmcp" est un pointeur pointant vers le premier maillon de la chaîne représentant une phrase à analyser.
"miny" et "maxy" sont des entiers appartenant à l'intervalle [1..25].

Résultat : /

Postcondition : AFFICHN3 affiche entre les lignes "miny" et "maxy", les mots correspondants aux informations enregistrées dans "noeudel.ensmots3el", cela, à partir de la liste de mots accessible par "firstmcp".

20. **TRSF**

cfr. 24. TRSF (p.87)

21. **FIND_INDEX**

cfr. 23. FIND_INDEX (p.87)

22. **VISUTXTPHR**

cfr. 8. VISUTXTPHR (p.77)

23. **MAJLIBRE**

Arguments : firstmcp
noeudel
remonte

Préconditions : "firstmcp" est un pointeur pointant vers le premier maillon de la chaîne contenant la phrase à analyser.
"noeudel" est un noeud d'analyse de type 'n1' élaboré par l'élève.
"remonte" est un booléen.

Résultat : /

Postconditions : MAJLIBRE met à jour en fonction de la valeur de "remonte", la mention 'libre' dans le maillon appartenant à la liste accessible par "firstmcp" et correspondant au mot choisi dans le noeud "noeudel" ("noeudel.mot1el").

24. **REMPLI_AM_SUBST_EL**

Arguments : noeudel
ligne
existam

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n2'.

"ligne" est un entier appartenant à l'intervalle [1..25].
"existam" est un booléen valant 'true' si le noeud "noeudel" contient déjà une analyse morphologique d'un substantif.

Résultat : noeudel

Postconditions : REMPLI_AM_SUBST_EL remplit la partie décrivant une analyse morphologique dans le noeud "noeudel" à partir des informations introduites par l'élève concernant l'analyse morphologique d'un substantif. Cette introduction sera réalisée à partir de questions posées à l'écran à partir de la ligne "ligne" jusqu'à celle "ligne + 4". Si "existam" vaut 'true', une réponse par défaut, en correspondance avec les informations de "noeudel", apparaîtra pour chaque question.

25. **REMPLI_AM_ADJ_EL**

Arguments : noeudel
ligne
existam

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n2'.
"ligne" est un entier appartenant à l'intervalle [1..25].
"existam" est un booléen valant 'true' si le noeud "noeudel" contient déjà une analyse morphologique d'un adjectif.

Résultat : noeudel

Postconditions : REMPLI_AM_ADJ_EL remplit la partie décrivant une analyse morphologique dans le noeud "noeudel" à partir des informations introduites par l'élève concernant l'analyse morphologique d'un adjectif. Cette introduction sera réalisée à partir de questions posées à l'écran à partir de la ligne "ligne" jusqu'à celle "ligne + 5".
Si "existam" vaut 'true', une réponse par défaut (en correspondance avec les informations de "noeudel") apparaîtra pour chaque question.

26. **REMPLI_AM_ADJPRON_EL**

Arguments : noeudel
ligne
existam

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n2'.
"ligne" est un entier appartenant à l'intervalle [1..25].
"existam" est un booléen valant 'true' si le noeud "noeudel" contient déjà une analyse morphologique d'un adjectif pronom.

Résultat : noeudel

Postconditions : REMPLI_AM_ADJPRON_EL remplit la partie décrivant une analyse morphologique dans le noeud "noeudel" à partir des informations introduites par l'élève concernant l'analyse morphologique d'un adjectif pronom. Cette introduction sera réalisée à partir de questions posées à l'écran à partir de la ligne "ligne" jusqu'à celle "ligne + 7". Si "existam" vaut 'true', une réponse par défaut, en correspondance avec les informations de "noeudel", apparaîtra pour chaque question.

27. **REMPLI_AM_V_EL**

Arguments : noeudel
ligne
existam

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n2'.
"ligne" est un entier appartenant à l'intervalle [1..25].
"existam" est un booléen valant 'true' si le noeud "noeudel" contient déjà une analyse morphologique d'un verbe.

Résultat : noeudel

Postconditions : REMPLI_AM_V_EL remplit la partie décrivant une analyse morphologique dans le noeud "noeudel" à partir des informations introduites par l'élève concernant l'analyse morphologique d'un verbe.

Cette introduction sera réalisée à partir de questions posées à l'écran à partir de la ligne "ligne" jusqu'à celle "ligne + 6". Si "existam" vaut 'true', une réponse par défaut, en correspondance avec les informations de "noeudel", apparaîtra pour chaque question.

28. **REMPLI_AM_PREP_EL**

Arguments : noeudel
ligne
existam

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n2'.
"ligne" est un entier appartenant à l'intervalle [1..25].
"existam" est un booléen valant 'true' si le noeud "noeudel" contient déjà une analyse morphologique d'une préposition.

Résultat : noeudel

Postconditions : REMPLI_AM_PREP_EL remplit la partie décrivant une analyse morphologique dans le noeud "noeudel" à partir des informations introduites par l'élève concernant l'analyse morphologique d'une préposition. Cette introduction sera réalisée à partir de questions posées à l'écran à partir de la ligne "ligne" jusqu'à celle "ligne + 2". Si "existam" vaut 'true', une réponse par défaut, en correspondance avec les informations de "noeudel", apparaîtra pour chaque question.

29. **REMPLI_AM_CONJ_EL**

Arguments : noeudel
ligne
existam

Préconditions : "noeudel" est un noeud d'analyse de l'élève de type 'n2'.
"ligne" est un entier appartenant à l'intervalle [1..25].

"existam" est un booléen valant 'true' si le noeud "noeudel" contient déjà une analyse morphologique d'une conjonction.

Résultat : noeudel

Postconditions : REMPLI_AM_CONJ_EL remplit la partie décrivant une analyse morphologique dans le noeud "noeudel" à partir des informations introduites par l'élève concernant l'analyse morphologique d'une conjonction. Cette introduction sera réalisée à partir de questions posées à l'écran à partir de la ligne "ligne" jusqu'à celle "ligne + 4". Si "existam" vaut 'true', une réponse par défaut, en correspondances avec les informations de "noeudel", apparaîtra pour chaque question.

30. **AFFICH_AM_SUBST_EL**

Arguments : noeudel
ligne

Préconditions : "noeudel" est un noeud d'analyse de type 'n2' élaboré par l'élève et contenant la description d'une analyse morphologique d'un substantif.
"ligne" est un entier appartenant à l'intervalle [1..25].

Résultat : /

Postconditions : AFFICH_AM_SUBST_EL affiche à partir de la ligne "ligne" et jusqu'à celle "ligne + 3", les informations grammaticales contenues dans "noeudel".

31. **AFFICH_AM_ADJ_EL**

Arguments : noeudel
ligne

Préconditions : "noeudel" est un noeud d'analyse de type 'n2' élaboré par l'élève et contenant la description d'une analyse morphologique d'un adjectif.
"ligne" est un entier appartenant à l'intervalle [1..25].

Résultat : /

Postconditions : AFFICH_AM_ADJ_EL affiche à partir de la ligne "ligne" et jusqu'à celle "ligne + 3", les informations grammaticales contenues dans "noeudel".

32. **AFFICH_AM_ADJPRON_EL**

Arguments : noeudel
ligne

Préconditions : "noeudel" est un noeud d'analyse de type 'n2' élaboré par l'élève et contenant la description d'une analyse morphologique d'un adjectif pronom.
"ligne" est un entier appartenant à l'intervalle [1..25].

Résultat : /

Postconditions : AFFICH_AM_ADJPRON_EL affiche à partir de la ligne "ligne" et jusqu'à celle "ligne + 3", les informations grammaticales contenues dans "noeudel".

33. **AFFICH_AM_V_EL**

Arguments : noeudel
ligne

Préconditions : "noeudel" est un noeud d'analyse de type 'n2' élaboré par l'élève et contenant la description d'une analyse morphologique d'un verbe.
"ligne" est un entier appartenant à l'intervalle [1..25].

Résultat : /

Postconditions : AFFICH_AM_V_EL affiche à partir de la ligne "ligne" et jusqu'à celle "ligne + 3", les informations grammaticales contenues dans "noeudel".

34. **AFFICH_AM_PREP_EL**

Arguments : noeudel
ligne

Préconditions : "noeudel" est un noeud d'analyse de type 'n2' élaboré par l'élève et contenant la description

d'une analyse morphologique d'une préposition.
"ligne" est un entier appartenant à l'intervalle [1..25].

Résultat : /

Postconditions : AFFICH_AM_PREP_EL affiche à partir de la ligne "ligne" et jusqu'à celle "ligne + 1", les informations grammaticales contenues dans "noeudel".

35. **AFFICH_AM_CONJ_EL**

Arguments : noeudel
ligne

Préconditions : "noeudel" est un noeud d'analyse de type 'n2' élaboré par l'élève et contenant la description d'une analyse morphologique d'une conjonction.
"ligne" est un entier appartenant à l'intervalle [1..25].

Résultat : /

Postconditions : AFFICH_AM_CONJ_EL affiche à partir de la ligne "ligne" et jusqu'à celle "ligne + 2", les informations grammaticales contenues dans "noeudel".

Remarque :

Toute personne intéressée par le code de chaque procédure, écrit en Turbo Pascal 4.0, aura recours aux annexes 16, 17, 18, 19, 20 et 21.

CONCLUSIONS

Nous avons vu que le domaine de recherche en E.A.O. n'est pas à négliger. En effet, les techniques actuellement utilisées pour permettre à l'enseignant de préparer, grâce à l'ordinateur, des leçons destinées à être proposées aux élèves, via cet ordinateur, ne satisfont pas toujours les utilisateurs.
Les raisons suivantes le démontrent. D'une part, nous avons expliqué, qu'en utilisant les langages auteur courants, le travail de préparation demandé au professeur pour créer les leçons est fort important. D'autre part, nous avons mentionné une technique permettant d'espérer la résolution de ces problèmes. Il s'agit du procédé faisant appel à l'intelligence artificielle et exploitant un modèle de connaissances de la matière enseignée et un modèle de l'élève, connus du système. Ce procédé n'est actuellement pas encore très répandu, son champ d'action se limite seulement à certains domaines privilégiés. Ce sont les difficultés de modélisation actuelles qui restreignent, pour l'instant, sa propagation. Ces problèmes surviennent au niveau du modèle de connaissances de la branche enseignée, mais, aussi et surtout, au niveau du modèle de l'apprenant.

Voilà pourquoi nous avons proposé dans ce mémoire, une technique nouvelle pouvant être qualifiée "d'intermédiaire" entre les deux procédés cités ci-dessus : "la technique semi-automatique".
Cette technique s'efforce de pallier les inconvénients des langages auteur actuels. Toutefois, ce système n'est pas aussi "intelligent" que celui exploité en I.C.A.L., car il ne contient pas de modèle de l'élève.
L'objectif de cette technique est de réduire le travail du professeur, tout en essayant d'améliorer la qualité des leçons obtenues. Pour cela, certaines informations typiques à la matière enseignée sont introduites dans le système. A partir de ces connaissances, le système peut générer automatiquement certaines données, introduites normalement par l'enseignant qui utilise la technique actuelle. Ce système permet à l'enseignant de décrire ses leçons d'une manière synthétique, en introduisant des données inconnues du système et nécessaires à celui-ci pour gérer la phase destinée à l'élève.

L'objectif premier de ce mémoire était de montrer l'efficacité de ce genre de méthode. Pour cela, nous avons voulu concrétiser cette idée et nous l'avons appliquée au latin, ou, plus exactement, aux exercices d'analyse et de traduction de textes latins.
Nous n'avons pas élaboré un didacticiel opérationnel, mais nous avons créé un outil de démonstration mettant en oeuvre la technique proposée.
Dans un but de facilité, pour nous permettre d'aboutir plus rapidemment à ce prototype, nous avons eu recours à l'aide du L.A.S.L.A. de l'Université de Liège pour certaines informations. Nous avons également posé certaines hypothèses restrictives, notamment quant à la structure d'une phrase latine traitée et quant à la gestion de sa traduction française.

Grâce aux connaissances latines introduites dans le système, ce dernier apporte, alors, une aide plus que non négligeable à l'enseignant, lors de la phase de préparation. Nous avons vu que l'intervention du professeur, au cours de cette phase, se situe à un autre niveau de détail que celui demandé par les langages auteur. Il doit, en effet, introduire des données inconnues du système, d'une part, pour compléter certaines étapes d'analyse et, d'autre part, pour aider le système à élaborer, lors de la "phase élève", le dialogue qu'il aura avec cet élève.

Disposant de l'outil de démonstration que nous avons élaboré, lors de ce travail, nous pouvons dire que la phase de préparation est fortement simplifiée pour l'enseignant grâce aux nombreuses aides du système. Il semble, en effet, que, pour élaborer la même leçon que celle obtenue par notre outil, le temps nécessaire à l'enseignant serait beaucoup plus important s'il utilisait un langage auteur.
Remarquons quand même que, dans notre travail, la liberté de l'enseignant quant au dialogue établi entre l'ordinateur et l'élève est réduite. Il n'a, en effet, pas la possibilité de choisir la disposition de l'écran ou la forme des questions. En revanche, il peut associer, à chaque étape d'analyse, un commentaire spécifique destiné à s'afficher à l'élève, à la place de celui du système. Il peut également choisir le type de chaque étape d'analyse et décider de l'affichage du commentaire à l'élève. Ces possibilités offertes à l'enseignant sont, nous semble-t-il, des décisions d'une importance beaucoup plus grande

que les premières non disponibles. Dès lors, ne vaut-il pas mieux ne pas pouvoir choisir, avec précision, la forme mais pouvoir décider du contenu, tout en disposant d'une aide appréciable du système, plutôt que de pouvoir (... ou devoir) tout déterminer seul !
Ce point de vue ne fait peut-être pas l'unanimité parmi les enseignants. Par la suite, il serait, alors, intéressant de prévoir, dans la phase de préparation, la possibilité pour le professeur d'indiquer quelles sont ses exigences quant à la "forme" de la leçon.
L'enseignant pourrait, dans ce cas, se soumettre entièrement aux décisions du système ou en déterminer certaines lui-même.

Les résultats obtenus grâce à notre "didacticiel de démonstration" semblent assez positifs et il est permis d'espérer des résultats futurs encore meilleurs. Il nous paraît, alors, intéressant d'envisager d'étendre notre produit de manière à le rendre opérationnel. Il est bien entendu que de nombreux développements pourront y être apportés, non seulement pour le rendre opérationnel, mais également pour l'optimiser.
Premièrement, il faudrait que ce logiciel devienne **indépendant** en ce qui concerne les informations fournies par le L.A.S.L.A. de l'Université de Liège. Pour aboutir à cette indépendance, il faudrait que le système soit capable d'élaborer lui-même les analyses morphologiques de chaque mot d'un texte quelconque. Cette aptitude lui permettrait d'étendre son champ d'action à tout texte latin, voir même à des textes créés par l'enseignant. Le didacticiel ne serait alors plus limité au traitement des seuls textes déjà analysés morphologiquement et jusqu'alors en provenance de Liège.
L'exigence d'un analyseur morphologique automatique fiable à 100% n'est pas obligatoire. Rappelons, d'ailleurs, comme nous l'avons mentionné au chapitre 7, que celui utilisé par le L.A.S.L.A. de Liège sur un gros système, ne fournit pas toujours tous les résultats demandés. Une solution intéressante a envisager serait l'élaboration d'un analyseur morphologique "semi-automatique". C'est-à-dire que, connaissant des règles grammaticales latines générales, il fournirait des résultats corrects dans la majorité des cas. L'enseignant interviendrait, ensuite, pour compléter les informations incorrectes ou manquantes.

Contrairement à notre travail, dans la situation où le système serait capable, seul ou aidé par le professeur, d'analyser morphologiquement une forme latine, le fait d'offrir à l'enseignant la possibilité de modifier le contenu du **lexique** aurait, alors, sa raison d'être. Ceci serait, en effet, intéressant puisque le système serait en mesure de traiter un quelconque nouveau mot latin.

Il faudrait, dès lors, peut-être dissocier les informations lexicales de celles morphologiques et ne plus tout concentrer dans un seul fichier, comme nous l'avons fait pour notre travail. Cette gestion du lexique, de même que celle du fichier "grammatical", comme nous l'avons souligné dans le chapitre 7 (7.1-Données initiales), soulèvent le problème de savoir s'il est préférable de créer un seul lexique général et une seule "grammaire" ou plusieurs, spécifiques chacun à un ou plusieurs texte(s) ou une disquette. Aucune de ces deux gestions extrêmes ne paraît cependant optimale !

Pour la gestion des informations lexicales, proposons une solution intermédiaire pouvant être valable quelle que soit la répartition des textes sur les disquettes. (Notons que cette solution pourrait aussi être envisageable pour les informations grammaticales).

Il s'agit, en fait, de combiner "lexique général" et "lexique spécifique". C'est-à-dire que, lors de la phase de préparation, l'enseignant disposerait d'un lexique général, figé et de lexiques spécifiques associés, chacun, à un texte traité. Chaque lexique spécifique contiendrait toutes les informations supplémentaires par rapport au lexique général et particulières au texte latin associé et analysé par l'enseignant. L'ensemble des données lexicales nécessaires à l'analyse d'un texte serait donc réparti entre le lexique spécifique et celui général.

La disquette destinée à l'élève contiendrait, outre, notamment, l'ensemble des textes mis à sa disposition, les informations lexicales y associées. C'est-à-dire chaque lexique spécifique utile et le morceau du lexique général contenant les informations nécessaires à tous les textes présents.

Cette solution semble être assez avantageuse aussi bien au niveau de la performance, que de la facilité et de la gestion de la place sur disquette.

Quoiqu'il en soit, quelle que soit la solution adoptée, l'enseignant aurait aussi la possibilité de visionner le contenu d'un ou du lexique latin-français. Il pourrait également y ajouter, supprimer ou modifier une ou plusieurs forme(s) latine(s) ainsi qu'une ou plusieurs traduction(s) y associée(s).

Des améliorations concernant, à présent, le **traitement automatique** géré par le système seraient aussi intéressantes à apporter au produit actuel.
A propos des vérifications "on-line" faites lors de l'analyse, le système pourrait notamment s'assurer du respect des conditions d'une analyse syntaxique correcte de la phrase latine traitée :
- tout mot de la phrase doit être associé à un symbole catégoriel syntaxique
- tout mot de la phrase ne peut être associé qu'à un seul symbole catégoriel syntaxique
- le mot de la phrase choisi comme étant l'élément central du groupe de mots associé à un symbole catégoriel syntaxique, doit appartenir à cet ensemble de mots
- lorsqu'une suite de mots est associée à un symbole catégoriel syntaxique, l'ordre des mots ne correspond pas obligatoirement à celui de la phrase. Ceci est dû au fait qu'en latin, la place des mots dans une phrase n'est pas fonctionnelle.

Au sujet du traitement automatique "off-line", vérifiant la cohérence de l'arbre d'analyse d'une phrase ou même de plusieurs phrases, son élaboration serait aussi à souhaiter, comme il en a été question au chapitre 7 (7.2.2- Spécification des procédures, 4. TRT_AUT_COH). Les renseignements fournis alors par le système permettrait au professeur de corriger d'éventuelles erreurs qu'il aurait commises, lors de ses compléments d'analyse.

De manière à améliorer la qualité des leçons produites, il est certain que les développements futurs doivent être établis en relation avec des enseignants, de préférence en langue latine.
Mais nous pouvons déjà penser à diverses améliorations pouvant être apportées à ce point de vue.
Premièrement, la gestion des **commentaires** associés à chaque étape d'analyse pourrait être étendue aussi bien pour les messages générés par le système que pour ceux ajoutés par l'enseignant.

A propos des commentaires introduits par l'enseignant, nous pourrions les considérer de plusieurs types :
- certains seraient "généraux", c'est-à-dire qu'ils seraient utilisables fréquemment, reprenant des règles standards (ex. "le verbe s'accorde avec le sujet", "AD exige l'accusatif", ...). Ces messages seraient en nombre limité et pourraient être mémorisés et facilement exploitables (ajout, suppression, sélection, modification).
- d'autres seraient plus spécifiques à tel ou tel cas précis survenant dans telle ou telle phrase. Ils ne seraient alors pas mémorisés puisque non standards. L'enseignant introduirait donc lui-même ces commentaires spécifiques chaque fois qu'ils seraient nécessaires.

Remarque :
Parmi les commentaires "généraux" enregistrés, certains pourraient être "à trous" de manière à être plus "personnalisés" dans chaque cas. Ces trous seraient à compléter autant que possible par le système, puis par l'enseignant.

Lors des compléments manuels de l'analyse, au moment où l'enseignant voudrait associer un commentaire à une étape de l'analyse, il aurait la possibilité d'indiquer s'il désire traiter un commentaire spécifique ou général.

Diverses solutions seraient à envisager pour gérer les commentaires "généraux" enregistrés dans le système :
soit accéder aux commentaires "généraux" par leur nom
soit accéder aux commentaires "généraux" par leur "contenu" : chaque commentaire s'afficherait l'un après l'autre à l'écran, afin que l'un d'eux puisse être sélectionné ou un autre ajouté
soit accéder aux commentaires "généraux" par leurs caractéristiques :
l'ensemble de ces commentaires formerait une BD. La sélection d'un commentaire se ferait en réduisant d'abord l'ensemble des messages en un sous-ensemble de commentaires répondants à certains critères : thème, caractéristiques, contexte, ... Cette réduction pourrait se faire en plusieurs étapes afin de limiter de plus en plus l'ensemble des commentaires valables. Une première sélection pourrait être établie automatiquement, puis, l'enseignant inter-

viendrait pour réduire encore ce sous-ensemble en donnant d'autres critères de sélection. Lorsque ce sous-ensemble de commentaires serait devenu relativement restreint, on pourrait envisager de visualiser tous ces messages afin de voir s'il en existe un que l'on pourrait associer à l'étape d'analyse que l'on veut commenter. Dans le cas négatif, s'il n'existait aucun commentaire répondant aux critères de sélection ou si aucun commentaire existant n'était choisi, il serait possible d'en ajouter un qui, bien évidemment, répondrait à toutes les conditions émises concernant ce message.

Critiques de ces propositions :

- La première solution ne serait pas évidente si le nombre de commentaires "généraux" était considérable :
comment se souvenir de tous les noms et les contenus ?
- La deuxième solution pourrait faire l'objet de la même critique que ci-dessus. De plus, elle serait plutôt lourde en cas de longs commentaires.
- La troisième solution pourrait être efficace, mais le problème serait de déterminer les caractéristiques identifiantes de ces commentaires.

Quelle que soit la solution adoptée, en plus du traitement de ces commentaires pour les associer à des textes, des phrases ou des étapes d'analyse, l'enseignant aurait la possibilité de gérer ces messages indépendamment de toute association. Ces messages seraient alors considérés comme de simples textes à créer, modifier ou supprimer.

Pour les messages générés automatiquement par le système, il serait aussi possible que le système génère, éventuellement à partir de "squelettes" de commentaires enregistrés, des messages individualisés à chaque cas. Cette individualisation des messages affichés les rendrait moins "froids" et plus "parlants", ce qui serait appréciable d'un point de vue pédagogique.
Concernant en particulier les commentaires apparaissant à l'élève lors d'une étape d'analyse destinée à établir une analyse morphologique, le système serait capable d'afficher des messages assez élaborés. Dans notre travail, nous nous sommes limités à l'affichage d'un message standard dans le cas de l'élaboration par l'élève d'un noeud d'analyse morphologique non prévu dans

l'arbre d'analyse associé. Cependant, grâce à ses informations morphologiques, le système pourrait examiner de plus près, lui-même, la réponse de l'élève. Il saurait automatiquement en détecter les éléments erronés. D'après ces erreurs, le système pourrait alors adapter son message de manière à essayer de faire découvrir à l'élève ses fautes. Ce genre de commentaire serait donc beaucoup plus "productif" d'un point de vue pédagogique qu'un simple avertissement mentionnant à l'élève l'inexactitude de sa réponse.

De manière à sensibiliser et stimuler davantage l'élève lors de ses exercices, un projet intéressant aussi à analyser serait la **personnalisation** des disquettes de travail des élèves.
Chaque élève aurait ses propres disquettes d'exercices. Celles-ci pourraient dès lors être configurables au niveau des messages à afficher au cours des exercices. Le logiciel deviendrait alors personnalisé et adapté à chaque élève. Pour cela, l'enseignant aurait accès à un fichier contenant divers messages (bienvenue, informations, ... sans oublier le message "vide") parmi lesquels il pourrait choisir ceux qui seraient le mieux appropriés à tel ou tel élève. Il est bien entendu que l'enseignant pourrait également non seulement modifier les messages contenus dans le fichier, mais aussi en ajouter d'autres inexistants.

Egalement dans le but d'adapter au mieux les exercices aux besoins des élèves, la gestion du **suivi** du travail de chaque élève pourrait être traitée. En effet, un tel suivi ne doit pas seulement être pris en compte dans un but de cotation. Il peut aussi fournir des informations à l'enseignant afin de l'aider à adapter au mieux ses cours aux problèmes fréquents des élèves.
Il serait également envisageable de donner à l'élève la possibilité d'introduire certains messages destinés à l'enseignant. De cette manière, l'élève pourrait faire savoir avec précision au professeur quels sont, par exemple, ses problèmes particuliers quant à l'exercice.

Tous ces développements apportés au produit actuel permettraient d'aboutir à un didacticiel, nous semble-t-il, assez efficace d'un point de vue pédagogique. Les qualités de ce logiciel seraient appréciées aussi bien par l'enseignant que par l'élève.
Par la suite, il serait, évidemment, intéressant d'appliquer la technique "semi-automatique" à d'autres matières enseignées.

BIBLIOGRAPHIE

Brousmiche J-P. (1987)
" Contributions d'un système de dialogue homme-machine à composante orale ", Mémoire de Licence et Maîtrise en Informatique, F.U.N.D.P., Namur, 1987.

Culley G.R. (1984)
" Teaching the Classics with Computers ", American Philological Association Commitee on Educational Services, Educational Papers 1, 1984.

Denooz J., Chef de travaux C.I.P.L., Université de Liège (1987)
" Didactique des Langues Anciennes et Ordinateur ",
" Faculté Ouverte ", Conférence Débats Dossiers,
Faculté de Philosophie et Lettres, Université de Liège, 1987.

Denooz J. (1976)
" Le Traitement des Textes Latins, Grecs et Français au Laboratoire d'Analyse Statistique des Langues Anciennes ",
Separata de la Revista de la Universidad Complutense,
Volumen XXV, numero 102, marzo-abril 1976.

Frippiat J. (1986)
" E.A.O. Réalisation d'un guide à la traduction de textes latins ou grecs. Analyse. ", Mémoire de licence et maîtrise en informatique, F.N.D.P., Namur.

Gaines B.R. (1982)
" Computers and People Series ", " Intelligent Tutoring Systems " edited by Sleeman D., University of Leeds, U.K. and Brown J.S., Cognitive and Instructional Sciences, Xerox Parc, U.S.A., Editions Academic Press, 1982.

Gouthière D.
" Pour un projet d'enseignement assisté par ordinateur sur l'accord du verbe ", Mémoire de Licence et Maîtrise en informatique, F.U.N.D.P., Namur.

Gross M. et Lentin A., Institut Blaise-Pascal (1967)
" Notions sur les Grammaires Formelles ", " Collection Programmation ", publiée sous la direction de Nolin L., Publications de l'Institut de Programmation de la Faculté des Sciences de Paris, Editions Gauthier-Villars, Paris, 1967.

Gross M. (1968)
" Langue et Langage : Grammaire Transformationnelle du Français : Syntaxe du Verbe ", Editions Larousse, Paris, 1968.

Haton J-P., CRIN-Université de Nancy I,
Perennou G., CERFIA-Université de Toulouse III (1978)
" Reconnaissance automatique de la parole ", " 8° Ecole d'Eté d'Informatique de l'AFCET ", Namur, juillet 1978.

Jobert G. et Perriault J. (1983)
" L'Enseignement Assisté par Ordinateur ",
numéro spécial de " Education Permanente ",
Editions Université de Paris-Dauphine, décembre 1983.

L.A.S.L.A.
" L'Ordinateur et le Latin : Techniques et Méthodes - Morphologie, Syntaxe et Lexicologie - Stylistique ",
Université de Liège.

Milner J-Cl. (1971)
" Aspects de la Théorie Syntaxique ", " L'Ordre Philosophique ",
collection dirigée par Ricoeur P. et Wahl F.,
Editions Du Seuil, Paris,
traduction de " Aspects of the Theory of Syntax ",
Chomsky N., 1965.

Paillet J-P. et Dugas A. (1977)
" Principes d'Analyse Syntaxique ", Editions Les Presses de l'Université du Québec, 1977, deuxième édition.

Richard Ph. et Ruwet N. (1968)
" L'Analyse Formelle des Langues Naturelles ",
Editions Mouton-Gauthier-Villars, Paris, 1968,
traduction de " Introduction to the Formal Analysis of Natural Languages ", Chomsky N. and Miller G.A.

Romainville M.
" Introduction aux logiciels auteur ",
" Formation et Recherche en Education " n° 1.12, Centre O.S.E.,
Département Education et Technologie, F.U.N.D.P. Namur.

Simon J-Cl. (1980)
" L'éducation et l'informatisation de la société ",
Rapport au Président de la République,
Annexes 1. : Les Voies de Développement et Contributions des
groupes de travail,
Editions La Documentation Française, Paris, 1980.

" Dix ans d'informatique dans l'enseignement secondaire :
1970-1980 ", " Recherches Pédagogiques ",
Institut National de Recherche Pédagogique.

" Le Micro-Professeur ", périodique trimestriel, volume I n° 4,
avril-sept. 1984.

" L'Enseignement des Langues par Ordinateur ",
" Le Micro-Professeur ", volume II n° 1, 1985, Liège.

" Le Latin et l'Enseignement Assisté par Ordinateur :
l'expérience de l'U.L.B. ", Université Libre de Bruxelles,
Faculté de Philosophie et Lettres, Groupe d'Informatique et de
Traitement Automatique, avril 1986.

Facultés Universitaires Notre-Dame de la Paix, Namur
Faculté d'Informatique
Année Académique 1987-1988

"Didacticiel d'Aide à l'Analyse et à la Traduction de Textes Latins"

Annexes

Sophie JACOB

Promoteur : Cl. Cherton

Mémoire présenté en vue de l'obtention du titre de Licenciée et Maître en Informatique

GUERRE DES GAULES

LIVRE PREMIER

La Gaule et ses habitants. I. L'ensemble de la Gaule est divisé en trois parties[1] : l'une est habitée par les Belges, l'autre par les Aquitains, la troisième par le peuple qui, dans sa langue, se nomme Celte, et, dans la nôtre, Gaulois. Tous ces peuples diffèrent entre eux par le langage, les coutumes, les lois[2]. Les Gaulois sont séparés des Aquitains par la Garonne, des Belges par la Marne et la Seine. Les plus braves de ces trois peuples sont les Belges, parce qu'ils sont les plus éloignés de la province romaine et des raffinements de sa civilisation, parce que les marchands y vont très rarement, et, par conséquent, n'y introduisent pas ce qui est propre à amollir les cœurs, enfin parce qu'ils sont les plus voisins des Germains, qui habitent sur l'autre rive du Rhin, et avec qui ils sont continuellement en guerre. C'est pour la même raison que les Helvètes aussi surpassent en valeur guerrière les autres Gaulois[3] : des combats presque quotidiens les mettent aux prises avec les Germains, soit qu'ils leur interdisent l'accès de leur terri-

1. César ne comprend pas sous le nom de *Gallia* la partie de la Gaule déjà soumise aux Romains.
2. La différence de langage entre les Belges et les Celtes n'était qu'une différence dialectale.
3. Les Helvètes, qui, au témoignage de Tacite, *Germ.*, 28,

BELLVM GALLICVM

LIBER PRIMVS

I. [1]Gallia est omnis diuisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur. [2]Hi omnes lingua, institutis, legibus inter se differunt. Gallos ab Aquitanis Garunna flumen, a Belgis Matrona et Sequana diuidit. [3]Horum omnium fortissimi sunt Belgae, propterea quod a cultu atque humanitate prouinciae longissime absunt, minimeque ad eos mercatores saepe commeant atque ea quae ad effeminandos animos pertinent inportant, proximique sunt Germanis, qui trans Rhenum incolunt, quibuscum continenter bellum gerunt. [4]Qua de causa Heluetii quoque reliquos Gallos uirtute praecedunt, quod fere cotidianis proeliis cum Germanis contendunt, cum aut suis finibus eos prohibent,

1. *Capita* I-XXX *usque ad* bello heluetiorum *in* S *desunt. Cap.* I-VI, 3 *usque ad* bono animo *in* L *a manu recentiori suppleta sunt, folio primo excisо.* 2 garunna BR β : -mna AM, N ‖ 4 heluetii : -itii A -idi M *[fere constanter]*.

toire, soit qu'ils les attaquent chez eux. La partie de la Gaule qu'occupent, comme nous l'avons dit, les Gaulois commence au Rhône, est bornée par la Garonne, l'Océan et la frontière de Belgique ; elle touche aussi au Rhin du côté des Séquanes et des Helvètes ; elle est orientée vers le nord. La Belgique commence où finit la Gaule[1] ; elle va jusqu'au cours inférieur du Rhin ; elle regarde vers le nord et vers l'est. L'Aquitaine s'étend de la Garonne aux Pyrénées et à la partie de l'Océan qui baigne l'Espagne ; elle est tournée vers le nord-ouest[2].

Les Helvètes. Plans ambitieux d'Orgétorix. Sa mort.

II. — Orgétorix était chez les Helvètes l'homme de beaucoup le plus distingué par la naissance et par la fortune. Sous le consulat de M. Messala et de M. Pison[3], séduit par le désir d'être roi, il forma une conspiration de la noblesse et persuada à ses concitoyens de sortir de leur pays avec la population entière : « Rien n'était plus facile, puisque leur valeur les mettait au-dessus de tous, que de devenir les maîtres de la Gaule entière. » Il eut d'autant moins de peine à les convaincre que les Helvètes, en raison des conditions géographiques, sont de toutes parts enfermés : d'un côté par le Rhin, dont le cours très large et très profond sépare l'Helvétie de la Germanie, d'un autre par le Jura, chaîne très haute qui se dresse entre les Helvètes et les Séquanes, et du troisième par le lac Léman et le Rhône, qui sépare notre province de leur territoire. Cela restreignait le champ de leurs courses vagabondes et les gênait pour porter la guerre chez leurs voisins : situation fort irritante pour des hommes qui avaient la passion de la guerre. Ils estimaient d'ailleurs que l'étendue de leur territoire, qui avait 240 milles de

avaient longtemps habité « entre la forêt hercynienne, le Rhin et le Mein », n'en étaient pas moins un peuple de race celtique.

1. Le mot *Gallia* est employé ici dans un sens restreint, et désigne seulement la Celtique, à l'exclusion de l'Aquitaine et de la Belgique. De même I, 44, 3, II, 3.1, III, 11, 3.

2. Sur ces erreurs d'orientation, voir l'*Introduction*, p. XIV

3. 61 av. J.-C. = 693 ab V. c.

aut ipsi in eorum finibus bellum ferunt. [5]Eorum una pars, quam Gallos optinere dictum est, initium capit a flumine Rhodano, continetur Garunna flumine, Oceano, finibus Belgarum, attingit etiam ab Sequanis et Heluetiis flumen Rhenum, uergit ad septentriones. [6]Belgae ab extremis Galliae finibus oriuntur, pertinent ad inferiorem partem fluminis Rheni, spectant in septentrionem et orientem solem. [7]Aquitania a Garunna flumine ad Pyrenaeos montes et eam partem Oceani quae est ad Hispaniam pertinet ; spectat inter occasum solis et septentriones.

II. [1]Apud Heluetios longe nobilissimus fuit et ditissimus Orgetorix. Is M. Messala [et P.] M. Pisone consulibus regni cupiditate inductus coniurationem nobilitatis fecit et ciuitati persuasit ut de finibus suis cum omnibus copiis exirent : [2]perfacile esse, cum uirtute omnibus praestarent, totius Galliae imperio potiri. [3]Id hoc facilius eis persuasit quod undique loci natura Heluetii continentur : una ex parte flumine Rheno latissimo atque altissimo, qui agrum Heluetium a Germanis diuidit ; altera ex parte monte Iura altissimo, qui est inter Sequanos et Heluetios ; tertia lacu Lemanno et flumine Rhodano, qui prouinciam nostram ab Heluetiis diuidit. [4]His rebus fiebat ut et minus late uagarentur et minus facile finitimis bellum inferre possent ; qua ex parte homines bellandi cupidi magno dolore adficiebantur. [5]Pro multitudine autem hominum et pro gloria belli atque fortitudinis angustos se fines

5-7 eorum una... et septentriones : *del. Bacher* ‖ oceano *Aldus* : -um *codd.* ‖ 6 septentrionem : -es *Aldus* ‖ 7 garunna : -mna T ‖ est α : *om.* β.

II. 1 et p : *del. Max Bonnet* ‖ 3 eis : iis *Spillmann* ‖ 3-4 una ex. adficiebantur : *del. Lange.*

LIVRE DEUXIÈME

Agitation en Belgique. I. César prenait ses quartiers d'hiver dans la Gaule citérieure, ainsi que nous l'avons dit plus haut, quand le bruit lui parvint à maintes reprises, confirmé par une lettre de Labiénus, que tous les peuples de la Belgique, qui forme, comme on l'a vu, un tiers de la Gaule, conspiraient contre Rome et échangeaient des otages. Les motifs du soulèvement étaient les suivants : d'abord, ils craignaient qu'une fois tout le reste de la Gaule pacifié nous ne menions contre eux nos troupes ; puis, un assez grand nombre de Gaulois les sollicitaient : les uns, de même qu'ils n'avaient pas voulu que les Germains s'attardassent en Gaule, supportaient mal de voir une armée romaine hiverner dans leur pays et s'y implanter ; les autres, en raison de la mobilité et de la légèreté de leur esprit, rêvaient de changer de maîtres ; ils recevaient aussi des avances de plusieurs personnages qui — le pouvoir se trouvant généralement en Gaule aux mains des puissants et des riches qui pouvaient acheter des hommes — arrivaient moins facilement à leurs fins sous notre domination[1].

1. Ce chapitre montre clairement que César avait décidé dès 58 d'établir la souveraineté de Rome sur la Gaule. L'hivernage des troupes en territoire séquane était l'affirmation audacieuse de ce dessein. Pour l'accomplir, il ajoutait à la force des armes l'action politique : il favorisait dans chaque cité le pouvoir régulier des magistrats et des sénats, contre les agitateurs ambitieux.

LIBER SECVNDVS

I. [1]Cum esset Caesar in citeriore Gallia, *legionesque essent conlocatae* in hibernis, ita uti supra demonstrauimus, crebri ad eum rumores adferebantur, litterisque item Labieni certior fiebat omnes Belgas, quam tertiam esse Galliae partem dixeramus, contra populum romanum coniurare obsidesque inter se dare. [2]Coniurandi has esse causas : primum quod uererentur ne omni pacata Gallia ad eos exercitus noster adduceretur ; [3]deinde, quod ab non nullis Gallis sollicitarentur, partim qui, ut Germanos diutius in Gallia uersari noluerant, ita populi romani exercitum hiemare atque inueterascere in Gallia moleste ferebant, partim qui mobilitate et leuitate animi nouis imperiis studebant, [4]ab non nullis etiam, quod in Gallia a potentioribus atque iis qui ad conducendos homines facultates habebant uulgo regna occupabantur, qui minus facile eam rem imperio nostro consequi poterant.

I. 1 <legionesque essent conlocatae> *add.* Klotz ‖ in hibernis : *om. f*, *Ul*, *del. Goeliitz* [*ol. R. E. A.*, 1925, *p.* 292]‖ supra : AM, BRS: superius LN *om.* β ‖ quam... dixeramus : *del. Schiller* ‖ 2 causas esse T/ ‖ 3 ut M, S : *om. cett.* ‖ noluerant α : nollent β ‖ 4 ab non nullis... occupabantur : *del. Nitsche* ‖ iis *ed. pr.* : his *codd.* ‖ imperio AM¹, SN β : in imperio M², BRL.

&GALLIA	NGALLIA	CE0010001001001111A00		001 0000100001
&GALLIA	NGALLIA	CE0010001001001111F00		001 0000100002
&EDO	2EST	CE0010001002002560C11		001 0000200001
&SVM	1EST	CE0010001002002560C11		001 0000200002
&SVM	2EST	CE0010001002002E6C11		001 0000200003
&OMNIS	OMNIS	CE0010001003003480A00	3	001 0000300001
&OMNIS	OMNIS	CE0010001003003480D00	1	001 0000300002
&OMNIS	OMNIS	CE0010001003003480L00	3	001 0000300003
&DIVIDO	DIVISA	CE0010001004004CA440	2	001 0000400001
&DIVIDO	DIVISA	CE0010001004004CF440	2	001 0000400002
&DIVIDO	DIVISA	CE0010001004004CJ440	6	001 0000400003
&DIVIDO	DIVISA	CE0010001004004CL440	6	001 0000400004
&DIVISVS	2DIVISA	CE0010001004004210A00	2	001 0000400005
&DIVISVS	2DIVISA	CE0010001004004210F00	2	001 0000400006
&DIVISVS	2DIVISA	CE0010001004004210J00	6	001 0000400007
&DIVISVS	2DIVISA	CE0010001004004210L00	6	001 0000400008
&IN	IN	CE0010001005005703000		001 0000500001
&IN	IN	CE0010001005005706000		001 0000500002
&PARS	PARTES	CE0010001006006130J00		001 0000600001
&PARS	PARTES	CE0010001006006130L00		001 0000600002
&TRES	TRES	CE0010001007007310J00	3	001 0000700001
&TRES	TRES	CE0010001007007310L00	3	001 0000700002
&QVI	1QUARUM	CE001000100800846M	2	001 0000800001
&QVIS	1QUARUM	CE001000100800847M	2	001 0000800002
&QVIS	2QUARUM	CE0010001008008480M00	2	001 0000800003
&VNVS	UNAM	CE0010001009009310C00	2	001 0000900001
&INCOLO	2INCOLUNT	CE0010001010010530L11		001 0001000001
&	BELGAE	CE0010001011011		001 0001100001
&ALIVS	ALIAM	CE0010001012012480C00	2	001 0001200001
&	AQUITANI	CE0010001013013		001 0001300001
&TERTIVS	TERTIAM	CE0010001014014320C00	2	001 0001400001
&QVI	1QUI	CE001000101501546A	4	001 0001500001
&QVI	1QUI	CE001000101501546J	4	001 0001500002
&QVI	2QUI	CE0010001015015660		001 0001500003
&QVI	3QUI	CE0010001015015670		001 0001500004
&QVI	4QUI	CE0010001015015600000		001 0001500005
&QVIS	1QUI	CE001000101501547A	4	001 0001500006
&QVIS	1QUI	CE001000101501547J	4	001 0001500007
&QVIS	2QUI	CE0010001015015480A00	4	001 0001500008
&QVIS	2QUI	CE0010001015015480J00	4	001 0001500009
&IPSE	IPSORUM	CE0010001016016450M00	5	001 0001600001
&LINGVA	LINGUA	CE0010001017017110A00		001 0001700001
&LINGVA	LINGUA	CE0010001017017110F00		001 0001700002
&	CELTAE	CE0010001018018		001 0001800001
&NOSTER	NOSTRA	CE0010001019019420A00	2	001 0001900001
&NOSTER	NOSTRA	CE0010001019019420F00	2	001 0001900002
&NOSTER	NOSTRA	CE0010001019019420J00	6	001 0001900003
&NOSTER	NOSTRA	CE0010001019019420L00	6	001 0001900004
&GALLI	NGALLI	CE0010001020020120J00		001 0002000001
&GALLVS	NGALLI	CE0010001020020210D00	5	001 0002000002
&GALLVS	NGALLI	CE0010001020020210J00	4	001 0002000003
&APPELLO	1APPELLANTUR	CE0010001021021510AL11		S001 0002100001
&APPELLO	2APPELLANTUR	CE0010001021021510CL31		S001 0002100002
&HIC	1HI	CE0010002001001450J00	4	001 0002200001
&OMNIS	OMNES	CE0010002002002480J00	3	001 0002300001
&OMNIS	OMNES	CE0010002002002480L00	3	001 0002300002
&LINGVA	LINGUA	CE0010002003003110A00		001 0002400001
&LINGVA	LINGUA	CE0010002003003110F00		001 0002400002
&INSTITVO	INSTITUTIS	CE0010002004004CN440	1	001 0002500001
&INSTITVO	INSTITUTIS	CE0010002004004CO440	1	001 0002500002
&INSTITVTVM	INSTITUTIS	CE0010002004004120N00		001 0002500003
&INSTITVTVM	INSTITUTIS	CE0010002004004120O00		001 0002500004
&LEX	LEGIBUS	CE0010002005005130N00		001 0002600001
&LEX	LEGIBUS	CE0010002005005130O00		001 0002600002
&INTER	INTER	CE0010002006006703000		001 0002700001

&SVI	1SE	CE001000200700743C00		001 0002800001
&SVI	1SE	CE001000200700743F00		001 0002800002
&DIFFERO	DIFFERUNT	CE001000200800856L11		2001 0002900001
&GALLI	NGALLOS	CE001000200900112L00		001 0003000001
&GALLVS	NGALLOS	CE001000200900121L00	4	001 0003000002
&AB	AB	CE001000201000270600		001 0003100001
&	AQUITANIS	CE0010002011003		001 0003200001
&	GARUNNA	CE0010002012004		001 0003300001
&FLVMEN	FLUMEN	CE001000201300513A00		001 0003400001
&FLVMEN	FLUMEN	CE001000201300513B00		001 0003400002
&FLVMEN	FLUMEN	CE001000201300513C00		001 0003400003
&AB	A	CE001000201400670600		001 0003500001
&AH	A	CE001000201400690000		001 0003500002
&	BELGIS	CE0010002015007		001 0003600001
&MATRONA	MATRONA	CE001000201600811A00		001 0003700001
&MATRONA	MATRONA	CE001000201600811F00		001 0003700002
&ET	1ET	CE001000201700960000		001 0003800001
&ET	2ET	CE001000201700981000		001 0003800002
&	SEQUANA	CE0010002018010		001 0003900001
&DIVIDO	DIVIDIT	CE001000201901153C11		S001 0004000001
&HIC	1HORUM	CE001000300100145M00	5	001 0004100001
&OMNIS	OMNIUM	CE001000300200248M00	1	001 0004200001
&FORTIS	FORTISSIMI	CE001000300300320D00	5	001 0004300001
&FORTIS	FORTISSIMI	CE001000300300320J00	4	001 0004300002
&SVM	1SUNT	CE001000300400456L11		001 0004400001
&SVM	2SUNT	CE001000300400456L11		001 0004400002
&	BELGAE	CE0010003005005		001 0004500001
&PROPTEREA	PROPTEREA	CE001000300600660000		001 0004600001
&QVI	1QUOD	CE001000300700746A	6	001 0004700001
&QVI	1QUOD	CE001000300700746C	6	001 0004700002
&QVIS	1QUOD	CE001000300700747A	6	001 0004700003
&QVIS	1QUOD	CE001000300700747C	6	001 0004700004
&QVIS	2QUOD	CE001000300700748A00	6	001 0004700005
&QVIS	2QUOD	CE001000300700748C00	6	001 0004700006
&QVOD	1QUOD	CE001000300700781000		001 0004700007
&QVOD	2QUOD	CE0010003007007820		001 0004700008
&AB	A	CE001000300800870600		001 0004800001
&AH	A	CE001000300800890000		001 0004800002
&COLO	2CULTU	CE001000300900953090		001 0004900001
&CVLTVS	1CULTU	CE001000300900914E00		001 0004900002
&CVLTVS	1CULTU	CE001000300900914F00		001 0004900003
&ATQVE	1ATQUE	CE001000301001081000		001 0005000001
&ATQVE	2ATQUE	CE0010003010010820		001 0005000002
&HVMANITAS	HUMANITATE	CE001000301101113F00		001 0005100001
&PROVINCIA	PROVINCIAE	CE001000301201211D00		001 0005200001
&PROVINCIA	PROVINCIAE	CE001000301201211E00		001 0005200002
&PROVINCIA	PROVINCIAE	CE001000301201211J00		001 0005200003
&LONGE	LONGISSIME	CE001000301301360-000		001 0005300001
&ABSVM	1ABSUNT	CE001000301401456L11		001 0005400001
&PARVM	2MINIME	CE001000301501560-000		001 0005500001
&QVE	QUE	CE001000301601681000		001 0005600001
&AD	AD	CE001000301701770300		001 0005700001
&IS	EOS	CE001000301801845L00	4	001 0005800001
&MERCATOR	MERCATORES	CE001000301901913J00		001 0005900001
&MERCATOR	MERCATORES	CE001000301901913L00		001 0005900002
&SAEPE	SAEPE	CE001000302002060000		001 0006000001
&SAEPES	SAEPE	CE001000302002013F00		001 0006000002
&COMMEO	COMMEANT	CE001000302102151L11		001 0006100001
&ATQVE	1ATQUE	CE001000302202281000		001 0006200001
&ATQVE	2ATQUE	CE0010003022022820		001 0006200002
&EA	EA	CE001000302302360000		001 0006300001
&IS	EA	CE001000302302345A00	2	001 0006300002
&IS	EA	CE001000302302345F00	2	001 0006300003
&IS	EA	CE001000302302345J00	6	001 0006300004
&IS	EA	CE001000302302345L00	6	001 0006300005

&QVI	1QUAE	CE0010003024024 46A	2	001 0006400001
&QVI	1QUAE	CE0010003024024 46J	2	001 0006400002
&QVI	1QUAE	CE0010003024024 46J	6	001 0006400003
&QVI	1QUAE	CE0010003024024 46L	6	001 0006400004
&QVIS	1QUAE	CE0010003024024 47A	2	001 0006400005
&QVIS	1QUAE	CE0010003024024 47J	2	001 0006400006
&QVIS	1QUAE	CE0010003024024 47J	6	001 0006400007
&QVIS	1QUAE	CE0010003024024 47L	6	001 0006400008
&QVIS	2QUAE	CE00100030240244 8A00	2	001 0006400009
&QVIS	2QUAE	CE00100030240244 8J00	2	001 0006400010
&QVIS	2QUAE	CE00100030240244 8J00	6	001 0006400011
&QVIS	2QUAE	CE00100030240244 8L00	6	001 0006400012
&AD	AD	CE0010003025025 70300		001 0006500001
&EFFEMINO	EFFEMINANDOS	CE0010003026026 5AL50	4	001 0006600001
&ANIMVS	ANIMOS	CE0010003027027 12L00		001 0006700001
&PERTINEO	PERTINENT	CE0010003028028 52L11		001 0006800001
&IMPORTO	IMPORTANT	CE0010003029029 51L11		001 0006900001
&PROPIOR	PROXIMI	CE0010003030030 2-D00	5	001 0007000001
&PROPIOR	PROXIMI	CE0010003030030 2-J00	4	001 0007000002
&PROXIMVM	PROXIMI	CE0010003030030 12D00		001 0007000003
&QVE	QUE	CE0010003031031 81000		001 0007100001
&SVM	1SUNT	CE0010003032032 56L11		001 0007200001
&SVM	2SUNT	CE0010003032032 E6L11		001 0007200002
&GERMANA	GERMANIS	CE0010003033033 11N00		001 0007300001
&GERMANA	GERMANIS	CE0010003033033 11000		001 0007300002
&GERMANVS	NGERMANIS	CE0010003033033 21N00	1	001 0007300003
&GERMANVS	NGERMANIS	CE0010003033033 21000	1	001 0007300004
&GERMANVS	1GERMANIS	CE0010003033033 12N00		001 0007300005
&GERMANVS	1GERMANIS	CE0010003033033 12000		001 0007300006
&GERMANVS	2GERMANIS	CE0010003033033 21N00	1	001 0007300007
&GERMANVS	2GERMANIS	CE0010003033033 21000	1	001 0007300008
&QVI	1QUI	CE0010003034034 46A	4	001 0007400001
&QVI	1QUI	CE0010003034034 46J	4	001 0007400002
&QVI	2QUI	CE0010003034034660		001 0007400003
&QVI	3QUI	CE0010003034034670		001 0007400004
&QVI	4QUI	CE0010003034034600 00		001 0007400005
&QVIS	1QUI	CE0010003034034 47A	4	001 0007400006
&QVIS	1QUI	CE0010003034034 47J	4	001 0007400007
&QVIS	2QUI	CE0010003034034 48A00	4	001 0007400008
&QVIS	2QUI	CE0010003034034 48J00	4	001 0007400009
&TRANS	TRANS	CE0010003035035 70300		001 0007500001
&	RHENUM	CE0010003036036		001 0007600001
&INCOLO	2INCOLUNT	CE0010003037037 53L11		001 0007700001
&QVI	1QUIBUS	CE0010003038038 46N	1	001 0007800001
&QVI	1QUIBUS	CE0010003038038 460	1	001 0007800002
&QVIS	1QUIBUS	CE0010003038038 47N	1	001 0007800003
&QVIS	1QUIBUS	CE0010003038038 470	1	001 0007800004
&QVIS	2QUIBUS	CE0010003038038 48N00	1	001 0007800005
&QVIS	2QUIBUS	CE0010003038038 48000	1	001 0007800006
&CVM	1CUM	CE0010003039039 60000		001 0007900001
&CVM	2CUM	CE0010003039039 70600		001 0007900002
&CVM	3CUM	CE0010003039039820		001 0007900003
&CONTINENTER	CONTINENTER	CE0010003040040 60000		001 0008000001
&BELLVM	BELLUM	CE0010003041041 12A00		001 0008100001
&BELLVM	BELLUM	CE0010003041041 12C00		001 0008100002
&BELLVS	BELLUM	CE0010003041041 21A00	6	001 0008100003
&BELLVS	BELLUM	CE0010003041041 21C00	5	001 0008100004
&GERO	GERUNT	CE0010003042042 53L11		S001 0008200001
&QVA	1QUA	CE0010004001001660		001 0008300001
&QVA	2QUA	CE0010004001001670		001 0008300002
&QVA	3QUA	CE0010004001001 60000		001 0008300003
&QVI	1QUA	CE0010004001001 46F	2	001 0008300004
&QVIS	1QUA	CE0010004001001 47F	2	001 0008300005
&QVIS	2QUA	CE0010004001001 48A00	2	001 0008300006
&QVIS	2QUA	CE0010004001001 48F00	2	001 0008300007

&QVIS	2QUA	CE00100040010014 8J00	6	001 0008300008
&QVIS	2QUA	CE00100040010014 8L00	6	001 0008300009
&DE	DE	CE00100040020027 0600		001 0008400001
&CAVSA	CAUSA	CE00100040030031 1A00		001 0008500001
&CAVSA	CAUSA	CE00100040030031 1F00		001 0008500002
&	HELVETII	CE0010004004004		001 0008600001
&QVISQVE	1QUOQUE	CE0010004005005 46F	5	001 0008700001
&QVISQVE	2QUOQUE	CE00100040050054 8F00	5	001 0008700002
&QVOQVE	QUOQUE	CE00100040050056 0000		001 0008700003
&RELIQVI	RELIQUOS	CE00100040060061 2L00		001 0008800001
&RELIQVVS	RELIQUOS	CE00100040060062 1L00	4	001 0008800002
&GALLI	NGALLOS	CE00100040070071 2L00		001 0008900001
&GALLVS	NGALLOS	CE00100040070072 1L00	4	001 0008900002
&VIRTVS	VIRTUTE	CE00100040080081 3F00		001 0009000001
&PRAECEDO	1PRAECEDUNT	CE00100040090095 3L11		001 0009100001
&QVI	1QUOD	CE0010004010010 46A	6	001 0009200001
&QVI	1QUOD	CE0010004010010 46C	6	001 0009200002
&QVIS	1QUOD	CE0010004010010 47A	6	001 0009200003
&QVIS	1QUOD	CE0010004010010 47C	6	001 0009200004
&QVIS	2QUOD	CE00100040100104 8A00	6	001 0009200005
&QVIS	2QUOD	CE00100040100104 8C00	6	001 0009200006
&QVOD	1QUOD	CE00100040100108 1000		001 0009200007
&QVOD	2QUOD	CE0010004010010820		001 0009200008
&FERE	FERE	CE00100040110116 0000		001 0009300001
&COTIDIANVS	COTIDIANIS	CE00100040120122 1N00	1	001 0009400001
&COTIDIANVS	COTIDIANIS	CE00100040120122 1O00	1	001 0009400002
&PROELIVM	PROELIIS	CE00100040130131 2N00		001 0009500001
&PROELIVM	PROELIIS	CE00100040130131 2O00		001 0009500002
&CVM	1CUM	CE00100040140146 0000		001 0009600001
&CVM	2CUM	CE00100040140147 0600		001 0009600002
&CVM	3CUM	CE0010004014014820		001 0009600003
&GERMANA	GERMANIS	CE00100040150151 1N00		001 0009700001
&GERMANA	GERMANIS	CE00100040150151 1O00		001 0009700002
&GERMANVS	NGERMANIS	CE00100040150152 1N00	1	001 0009700003
&GERMANVS	NGERMANIS	CE00100040150152 1O00	1	001 0009700004
&GERMANVS	1GERMANIS	CE00100040150151 2N00		001 0009700005
&GERMANVS	1GERMANIS	CE00100040150151 2O00		001 0009700006
&GERMANVS	2GERMANIS	CE00100040150152 1N00	1	001 0009700007
&GERMANVS	2GERMANIS	CE00100040150152 1O00	1	001 0009700008
&CONTENDO	CONTENDUNT	CE00100040160165 3L11		001 0009800001
&CVM	1CUM	CE00100040170176 0000		001 0009900001
&CVM	2CUM	CE00100040170177 0600		001 0009900002
&CVM	3CUM	CE0010004017017820		001 0009900003
&AVT	AUT	CE00100040180188 1000		001 0010000001
&SVI	2SUIS	CE00100040190194 4N00	1	001 0010100001
&SVI	2SUIS	CE00100040190194 4O00	1	001 0010100002
&SVO	SUIS	CE00100040190195 3B11		001 0010100003
&SVS	SUIS	CE00100040190191 3D00		001 0010100004
&SVVM	SUIS	CE00100040190191 2N00		001 0010100005
&SVVM	SUIS	CE00100040190191 2O00		001 0010100006
&SVVS	SUIS	CE00100040190194 4N00	1	001 0010100007
&SVVS	SUIS	CE00100040190194 4O00	1	001 0010100008
&FINIS	FINIBUS	CE00100040200201 3N00		001 0010200001
&FINIS	FINIBUS	CE00100040200201 3O00		001 0010200002
&IS	EOS	CE00100040210214 5L00	4	001 0010300001
&PROHIBEO	PROHIBENT	CE00100040220225 2L11		001 0010400001
&AVT	AUT	CE00100040230238 1000		001 0010500001
&IPSE	IPSI	CE00100040240244 5E00	1	001 0010600001
&IPSE	IPSI	CE00100040240244 5J00	4	001 0010600002
&IN	IN	CE00100040250257 0300		001 0010700001
&IN	IN	CE00100040250257 0600		001 0010700002
&IS	EORUM	CE00100040260264 5M00	5	001 0010800001
&FINIS	FINIBUS	CE00100040270271 3N00		001 0010900001
&FINIS	FINIBUS	CE00100040270271 3O00		001 0010900002
&BELLVM	BELLUM	CE00100040280281 2A00		001 0011000001

&BELLVM	BELLUM	CE001000402802812C00		001 0011000002
&BELLVS	BELLUM	CE001000402802821A00	6	001 0011000003
&BELLVS	BELLUM	CE001000402802821C00	5	001 0011000004
&GERO	GERUNT	CE001000402902953L11		S001 0011100001
&IS	EORUM	CE001000500100145M00	5	001 0011200001
&VNA	UNA	CE001000500200260000		001 0011300001
&VNVS	UNA	CE001000500200231A00	2	001 0011300002
&VNVS	UNA	CE001000500200231F00	2	001 0011300003
&VNVS	UNA	CE001000500200231J00	6	001 0011300004
&VNVS	UNA	CE001000500200231L00	6	001 0011300005
&PARS	PARS	CE001000500300313A00		001 0011400001
&PARS	PARS	CE001000500300313B00		001 0011400002
&QVAM	1QUAM	CE0010005004004660		001 0011500001
&QVAM	2QUAM	CE0010005004004670		001 0011500002
&QVI	1QUAM	CE0010005004004460	2	001 0011500003
&QVIS	1QUAM	CE0010005004004470	2	001 0011500004
&QVIS	2QUAM	CE001000500400448C00	2	001 0011500005
&GALLI	NGALLOS	CE001000500500512L00		001 0011600001
&GALLVS	NGALLOS	CE001000500500521L00	4	001 0011600002
&OBTINEO	OBTINERE	CE001000500600652071		001 0011700001
&DICO	2DICTUM	CE001000500700750A44	6	001 0011800001
&DICO	2DICTUM	CE001000500700750C44	5	001 0011800002
&DICO	2DICTUM	CE001000500700753080		001 0011800003
&DICTVM	DICTUM	CE001000500700712A00		001 0011800004
&DICTVM	DICTUM	CE001000500700712C00		001 0011800005
&EDO	2EST	CE0010005008008560011		001 0011900001
&SVM	1EST	CE0010005008008560011		001 0011900002
&SVM	2EST	CE0010005008008E6C11		001 0011900003
&INITIVM	INITIUM	CE001000500900912A00		001 0012000001
&INITIVM	INITIUM	CE001000500900912C00		001 0012000002
&CAPIO	2CAPIT	CE0010005010010550011		001 0012100001
&AB	A	CE001000501101170600		001 0012200001
&AH	A	CE001000501101190000		001 0012200002
&FLVMEN	FLUMINE	CE001000501201213F00		001 0012300001
&	RHODANO	CE0010005013013		001 0012400001
&CONTINEO	CONTINETUR	CE001000501401458C11		001 0012500001
&	GARUNNA	CE0010005015015		001 0012600001
&FLVMEN	FLUMINE	CE001000501601613F00		001 0012700001
&OCEANVS	NOCEANO	CE001000501701712E00		001 0012800001
&OCEANVS	NOCEANO	CE001000501701712F00		001 0012800002
&OCEANVS	1OCEANO	CE001000501701712E00		001 0012800003
&OCEANVS	1OCEANO	CE001000501701712F00		001 0012800004
&OCEANVS	2OCEANO	CE001000501701721E00	5	001 0012800005
&OCEANVS	2OCEANO	CE001000501701721F00	5	001 0012800006
&FINIS	FINIBUS	CE001000501801813N00		001 0012900001
&FINIS	FINIBUS	CE001000501801813O00		001 0012900002
&	BELGARUM	CE0010005019019		001 0013000001
&ATTINGO	ATTINGIT	CE001000502002053C11		001 0013100001
&ETIAM	ETIAM	CE001000502102160000		001 0013200001
&AB	AB	CE001000502202270600		001 0013300001
&	SEQUANIS	CE0010005023023		001 0013400001
&ET	1ET	CE001000502402460000		001 0013500001
&ET	2ET	CE001000502402481000		001 0013500002
&	HELVETIIS	CE0010005025025		001 0013600001
&FLVMEN	FLUMEN	CE001000502602613A00		001 0013700001
&FLVMEN	FLUMEN	CE001000502602613B00		001 0013700002
&FLVMEN	FLUMEN	CE001000502602613C00		001 0013700003
&	RHENUM	CE0010005027027		001 0013800001
&VERGO	VERGIT	CE001000502802853C11		001 0013900001
&AD	AD	CE001000502902970300		001 0014000001
&SEPTENTRIO	SEPTENTRIONES	CE001000503003013J00		S001 0014100001
&SEPTENTRIO	SEPTENTRIONES	CE001000503003013L00		S001 0014100002
&	BELGAE	CE0010006001001		001 0014200001
&AB	AB	CE001000600200270600		001 0014300001
&EXTER	EXTREMIS	CE001000600300320N00	1	001 0014400001

&EXTER	EXTREMIS	CE001000600300 32J000	1	001 0014400002
&EXTREMA	EXTREMIS	CE0010006003 00312N00		001 0014400003
&EXTREMA	EXTREMIS	CE0010006003 00312000		001 0014400004
&EXTREMVM	EXTREMIS	CE0010006003 00312N00		001 0014400005
&EXTREMVM	EXTREMIS	CE0010006003 00312000		001 0014400006
&GALLIA	NGALLIAE	CE0010006004 00411D00		001 0014500001
&GALLIA	NGALLIAE	CE0010006004 00411E00		001 0014500002
&GALLIA	NGALLIAE	CE0010006004 00411J00		001 0014500003
&FINIS	FINIBUS	CE0010006005 00513N00		001 0014600001
&FINIS	FINIBUS	CE0010006005 00513000		001 0014600002
&ORIOR	ORIUNTUR	CE0010006006 0065ML11		001 0014700001
&PERTINEO	PERTINENT	CE0010006007 00752L11		001 0014800001
&AD	AD	CE0010006008 00870300		001 0014900001
&INFERVS	INFERIOREM	CE0010006009 0092AC00	3	001 0015000001
&PARS	PARTEM	CE0010006010 01013C00		001 0015100001
&FLVMEN	FLUMINIS	CE0010006011 01113D00		001 0015200001
&	RHENI	CE0010006012012		001 0015300001
&SPECTO	SPECTANT	CE0010006013 01351L11		001 0015400001
&IN	IN	CE0010006014 01470300		001 0015500001
&IN	IN	CE0010006014 01470600		001 0015500002
&SEPTENTRIO	SEPTENTRIONEM	CE0010006015 01513C00		001 0015600001
&ET	1ET	CE0010006016 01660000		001 0015700001
&ET	2ET	CE0010006016 01681000		001 0015700002
&ORIENS	ORIENTEM	CE0010006017 01713C00		001 0015800001
&ORIENS	NORIENTEM	CE0010006017 01713C00		001 0015800002
&ORIOR	ORIENTEM	CE0010006017 0175MC41	3	001 0015800003
&SOL	SOLEM	CE0010006018 01813C00		S001 0015900001
&	AQUITANIA	CE0010007001001		001 0016000001
&AB	A	CE0010007002 00270600		001 0016100001
&AH	A	CE0010007002 00290000		001 0016100002
&	GARUNNA	CE0010007003003		001 0016200001
&FLVMEN	FLUMINE	CE0010007004 00413F00		001 0016300001
&AD	AD	CE0010007005 00570300		001 0016400001
&PYRENAEVS	NPYRENAEOS	CE0010007006 00621L00	4	001 0016500001
&MONS	MONTES	CE0010007007 00713J00		001 0016600001
&MONS	MONTES	CE0010007007 00713L00		001 0016600002
&ET	1ET	CE0010007008 00860000		001 0016700001
&ET	2ET	CE0010007008 00881000		001 0016700002
&EO	1EAM	CE0010007009 00956A31		001 0016800001
&IS	EAM	CE0010007009 00945C00	2	001 0016800002
&PARS	PARTEM	CE0010007010 01013C00		001 0016900001
&OCEANVS	NOCEANI	CE0010007011 01112D00		001 0017000001
&OCEANVS	NOCEANI	CE0010007011 01112J00		001 0017000002
&OCEANVS	1OCEANI	CE0010007011 01112D00		001 0017000003
&OCEANVS	1OCEANI	CE0010007011 01112J00		001 0017000004
&OCEANVS	2OCEANI	CE0010007011 01121D00	5	001 0017000005
&OCEANVS	2OCEANI	CE0010007011 01121J00	4	001 0017000006
&QVI	1QUAE	CE0010007012 01246A	2	001 0017100001
&QVI	1QUAE	CE0010007012 01246J	2	001 0017100002
&QVI	1QUAE	CE0010007012 01246J	6	001 0017100003
&QVI	1QUAE	CE0010007012 01246L	6	001 0017100004
&QVIS	1QUAE	CE0010007012 01247A	2	001 0017100005
&QVIS	1QUAE	CE0010007012 01247J	2	001 0017100006
&QVIS	1QUAE	CE0010007012 01247J	6	001 0017100007
&QVIS	1QUAE	CE0010007012 01247L	6	001 0017100008
&QVIS	2QUAE	CE0010007012 01248A00	2	001 0017100009
&QVIS	2QUAE	CE0010007012 01248J00	2	001 0017100010
&QVIS	2QUAE	CE0010007012 01248J00	6	001 0017100011
&QVIS	2QUAE	CE0010007012 01248L00	6	001 0017100012
&EDO	2EST	CE0010007013 01356C11		001 0017200001
&SVM	1EST	CE0010007013 01356C11		001 0017200002
&SVM	2EST	CE0010007013 013E6C11		001 0017200003
&AD	AD	CE0010007014 01470300		001 0017300001
&HISPANIA	NHISPANIAM	CE0010007015 01511C00		001 0017400001
&PERTINEO	PERTINET	CE0010007016 01652C11		001 0017500001

&SPECTO	SPECTAT	CE001000701701751C11		001 0017600001
&INTER	INTER	CE001000701801870300		001 0017700001
&OCCASVS	OCCASUM	CE001000701901914C00		001 0017800001
&OCCIDO	1OCCASUM	CE001000701901195CA44	6	001 0017800002
&OCCIDO	1OCCASUM	CE001000701901195CC44	5	001 0017800003
&OCCIDO	1OCCASUM	CE001000701901953080		001 0017800004
&SOL	SOLIS	CE001000702002013D00		001 0017900001
&SOLVM	1SOLIS	CE001000702002012N00		001 0017900002
&SOLVM	1SOLIS	CE001000702002012000		001 0017900003
&SOLVS	SOLIS	CE001000702002048N00	1	001 0017900004
&SOLVS	SOLIS	CE001000702002048000	1	001 0017900005
&ET	1ET	CE001000702102160000		001 0018000001
&ET	2ET	CE001000702102181000		001 0018000002
&SEPTENTRIO	SEPTENTRIONES	CE001000702202213J00		K001 0018100001
&SEPTENTRIO	SEPTENTRIONES	CE001000702202213L00		K001 0018100002

&CVM	1CUM	CE0010001001001600000		002 0000100001
&CVM	2CUM	CE0010001001001706000		002 0000100002
&CVM	3CUM	CE0010001001001820		002 0000100003
&EDO	2ESSET	CE0010001002002056C32		002 0000200001
&SVM	1ESSET	CE0010001002002056C32		002 0000200002
&SVM	2ESSET	CE0010001002002E6C32		002 0000200003
&CAESAR	NCAESAR	CE0010001003003130A00		002 0000300001
&CAESAR	NCAESAR	CE0010001003003130B00		002 0000300002
&IN	IN	CE0010001004004703000		002 0000400001
&IN	IN	CE0010001004004706000		002 0000400002
&CITERIOR	CITERIORE	CE0010001005005280F00	1	002 0000500001
&GALLIA	NGALLIA	CE0010001006006110A00		002 0000600001
&GALLIA	NGALLIA	CE0010001006006110F00		002 0000600002
&ITA	ITA	CE0010001007007600000		002 0000700001
&VT	1UTI	CE0010001008008660		002 0000800001
&VT	2UTI	CE0010001008008670		002 0000800002
&VT	3UTI	CE0010001008008600000		002 0000800003
&VT	4UTI	CE0010001008008820		002 0000800004
&VTOR	UTI	CE0010001008008SL071		002 0000800005
&SVPRA	1SUPRA	CE0010001009009600000		002 0000900001
&SVPRA	2SUPRA	CE0010001009009703000		002 0000900002
&DEMONSTRO	DEMONSTRAVIMUS	CE0010001010010511J14		002 0001000001
&CREBER	CREBRI	CE0010001011011210D00	5	002 0001100001
&CREBER	CREBRI	CE0010001011011210J00	4	002 0001100002
&AD	AD	CE0010001012012703000		002 0001200001
&IS	EUM	CE0010001013013450C00	4	002 0001300001
&RVMOR	RUMORES	CE0010001014014130J00		002 0001400001
&RVMOR	RUMORES	CE0010001014014130L00		002 0001400002
&AFFERO	ADFEREBANTUR	CE0010001015015SFL12		002 0001500001
&LITTERA	LITTERIS	CE0010001016016110N00		002 0001600001
&LITTERA	LITTERIS	CE0010001016016110O0		002 0001600002
&QVE	QUE	CE0010001017017810O0		002 0001700001
&ITEM	ITEM	CE0010001018018600000		002 0001800001
&LABIENVS	NLABIENI	CE0010001019019120D00		002 0001900001
&LABIENVS	NLABIENI	CE0010001019019120J00		002 0001900002
&CERTVS	CERTIOR	CE0010001020020AA00	3	002 0002000001
&FIO	FIEBAT	CE0010001021021560C12		002 0002100001
&OMNIS	OMNES	CE0010001022022480J00	3	002 0002200001
&OMNIS	OMNES	CE0010001022022480L00	3	002 0002200002
&	BELGAS	CE0010001023023		002 0002300001
&QVAM	1QUAM	CE0010001024024660		002 0002400001
&QVAM	2QUAM	CE0010001024024670		002 0002400002
&QVI	1QUAM	CE0010001024024460	2	002 0002400003
&QVIS	1QUAM	CE0010001024024470	2	002 0002400004
&QVIS	2QUAM	CE0010001024024480O0	2	002 0002400005
&TERTIVS	TERTIAM	CE0010001025025320C00	2	002 0002500001
&EDO	2ESSE	CE0010001026026560710		002 0002600001
&SVM	1ESSE	CE0010001026026560710		002 0002600002
&SVM	2ESSE	CE0010001026026E60710		002 0002600003
&GALLIA	NGALLIAE	CE0010001027027110D00		002 0002700001
&GALLIA	NGALLIAE	CE0010001027027110E00		002 0002700002
&GALLIA	NGALLIAE	CE0010001027027110J00		002 0002700003
&PARS	PARTEM	CE0010001028028130C00		002 0002800001
&DICO	2DIXERAMUS	CE0010001029029530J15		002 0002900001
&CONTRA	1CONTRA	CE0010001030030600000		002 0003000001
&CONTRA	2CONTRA	CE0010001030030703000		002 0003000002
&POPVLVS	1POPULUM	CE0010001031031120C00		002 0003100001
&POPVLVS	2POPULUM	CE0010001031031120C00		002 0003100002
&ROMANVS	NROMANUM	CE0010001032032210A00	6	002 0003200001
&ROMANVS	NROMANUM	CE0010001032032210C00	5	002 0003200002
&CONIVRO	CONIURARE	CE0010001033033510710		002 0003300001
&OBSES	OBSIDES	CE0010001034034130J00		002 0003400001
&OBSES	OBSIDES	CE0010001034034130L00		002 0003400002

&OBSIDEO	OBSIDES	CE001000103403452B11		002 0003400003
&OBSIDO	OBSIDES	CE001000103403453B13		002 0003400004
&QVE	QUE	CE001000103503581000		002 0003500001
&INTER	INTER	CE001000103603670300		002 0003600001
&SVI	1SE	CE001000103703743C00		002 0003700001
&SVI	1SE	CE001000103703743F00		002 0003700002
&DO	DARE	CE001000103803851071		S002 0003800001
&CONIVRO	CONIURANDI	CE00100020010015AD50	5	002 0003900001
&CONIVRO	CONIURANDI	CE00100020010015AJ50	4	002 0003900002
&CONIVRO	CONIURANDI	CE001000200100151D60		002 0003900003
&HIC	1HAS	CE001000200200245L00	2	002 0004000001
&EDO	2ESSE	CE001000200300356071		002 0004100001
&SVM	1ESSE	CE001000200300356071		002 0004100002
&SVM	2ESSE	CE00100020003003E6071		002 0004100003
&CAVSA	CAUSAS	CE001000200400411L00		002 0004200001
&PRIMVM	PRIMUM	CE00100020050053N000		002 0004300001
&PRIMVS	PRIMUM	CE00100020050053KA00	6	002 0004300002
&PRIMVS	PRIMUM	CE00100020050053KC00	5	002 0004300003
&QVI	1QUOD	CE0010000200600646A	6	002 0004400001
&QVI	1QUOD	CE0010000200600646C	6	002 0004400002
&QVIS	1QUOD	CE0010000200600647A	6	002 0004400003
&QVIS	1QUOD	CE0010000200600647C	6	002 0004400004
&QVIS	2QUOD	CE001000200600648A00	6	002 0004400005
&QVIS	2QUOD	CE001000200600648C00	6	002 0004400006
&QVOD	1QUOD	CE001000200600681000		002 0004400007
&QVOD	2QUOD	CE00100002006006820		002 0004400008
&VEREOR	VERERENTUR	CE00100020070075KL32		002 0004500001
&NE	1NE	CE001000200800860000		002 0004600001
&NE	2NE	CE00100002008008670		002 0004600002
&NE	3NE	CE001000200800868000		002 0004600003
&NE	4NE	CE00100002008008820		002 0004600004
&OMNIS	OMNI	CE001000200900948E00	1	002 0004700001
&OMNIS	OMNI	CE001000200900948F00	1	002 0004700002
&PACATVS	PACATA	CE001000201001021A00	2	002 0004800001
&PACATVS	PACATA	CE001000201001021F00	2	002 0004800002
&PACATVS	PACATA	CE001000201001021J00	6	002 0004800003
&PACATVS	PACATA	CE001000201001021L00	6	002 0004800004
&PACO	PACATA	CE00100020100105AA44	2	002 0004800005
&PACO	PACATA	CE00100020100105AF44	2	002 0004800006
&PACO	PACATA	CE00100020100105AJ44	6	002 0004800007
&PACO	PACATA	CE00100020100105AL44	6	002 0004800008
&GALLIA	NGALLIA	CE001000201101111A00		002 0004900001
&GALLIA	NGALLIA	CE001000201101111F00		002 0004900002
&AD	AD	CE001000201201270300		002 0005000001
&IS	EOS	CE001000201301345L00	4	002 0005100001
&EXERCEO	EXERCITUS	CE00100020140145BA44	4	002 0005200001
&EXERCITVS	1EXERCITUS	CE001000201401414A00		002 0005200002
&EXERCITVS	1EXERCITUS	CE001000201401414D00		002 0005200003
&EXERCITVS	1EXERCITUS	CE001000201401414J00		002 0005200004
&EXERCITVS	1EXERCITUS	CE001000201401414L00		002 0005200005
&EXERCITVS	2EXERCITUS	CE001000201401421A00	4	002 0005200006
&NOSTER	NOSTER	CE001000201501542A00	4	002 0005300001
&ADDVCO	ADDUCERETUR	CE00100020160165CC32		0002 0005400001
&DEINDE	DEINDE	CE001000300101760000		002 0005500001
&QVI	1QUOD	CE0010000300201846A	6	002 0005600001
&QVI	1QUOD	CE0010000300201846C	6	002 0005600002
&QVIS	1QUOD	CE0010000300201847A	6	002 0005600003
&QVIS	1QUOD	CE0010000300201847C	6	002 0005600004
&QVIS	2QUOD	CE001000300201848A00	6	002 0005600005
&QVIS	2QUOD	CE001000300201848C00	6	002 0005600006
&QVOD	1QUOD	CE001000300201881000		002 0005600007
&QVOD	2QUOD	CE00100003002018820		002 0005600008
&AB	AB	CE001000300301970600		002 0005700001

&NONNVLLVS	NONNULLIS	CE001000300402048N00	1	002 0005800001
&NONNVLLVS	NONNULLIS	CE001000300402048000	1	002 0005800002
&GALLI	NGALLIS	CE001000300502112N00		002 0005900001
&GALLI	NGALLIS	CE001000300502112000		002 0005900002
&GALLVS	NGALLIS	CE001000300502121N00	1	002 0005900003
&GALLVS	NGALLIS	CE001000300502121000	1	002 0005900004
&SOLLICITO	SOLLICITARENTUR	CE00100030060225AL32		002 0006000001
&PARTIM	PARTIM	CE001000300702360000		002 0006100001
&QVI	1QUI	CE001000300802446A	4	002 0006200001
&QVI	1QUI	CE001000300802446J	4	002 0006200002
&QVI	2QUI	CE0010003008024660		002 0006200003
&QVI	3QUI	CE0010003008024670		002 0006200004
&QVI	4QUI	CE001000300802460000		002 0006200005
&QVIS	1QUI	CE001000300802447A	4	002 0006200006
&QVIS	1QUI	CE001000300802447J	4	002 0006200007
&QVIS	2QUI	CE001000300802448A00	4	002 0006200008
&QVIS	2QUI	CE001000300802448J00	4	002 0006200009
&VT	1UT	CE0010003009025660		002 0006300001
&VT	2UT	CE0010003009025670		002 0006300002
&VT	3UT	CE001000300902560000		002 0006300003
&VT	4UT	CE0010003009025820		002 0006300004
&GERMANVS	NGERMANOS	CE001000301002621L00	4	002 0006400001
&GERMANVS	1GERMANOS	CE001000301002612L00		002 0006400002
&GERMANVS	2GERMANOS	CE001000301002621L00	4	002 0006400003
&DIV	DIUTIUS	CE00100030110276&000		002 0006500001
&IN	IN	CE001000301202870300		002 0006600001
&IN	IN	CE001000301202870600		002 0006600002
&GALLIA	NGALLIA	CE001000301302911A00		002 0006700001
&GALLIA	NGALLIA	CE001000301302911F00		002 0006700002
&VERSO	VERSARI	CE00100030140305A071		002 0006800001
&VERSOR	VERSARI	CE00100030140305J071		002 0006800002
&NOLO	NOLUERANT	CE001000301503156L15		002 0006900001
&ITA	ITA	CE001000301603260000		002 0007000001
&POPVLVS	1POPULI	CE001000301703312D00		002 0007100001
&POPVLVS	1POPULI	CE001000301703312J00		002 0007100002
&POPVLVS	2POPULI	CE001000301703312D00		002 0007100003
&POPVLVS	2POPULI	CE001000301703312J00		002 0007100004
&ROMANI	NROMANI	CE001000301803412J00		002 0007200001
&ROMANVS	NROMANI	CE001000301803421D00	5	002 0007200002
&ROMANVS	NROMANI	CE001000301803421J00	4	002 0007200003
&EXERCEO	EXERCITUM	CE00100030190355BA44	6	002 0007300001
&EXERCEO	EXERCITUM	CE00100030190355BC44	5	002 0007300002
&EXERCEO	EXERCITUM	CE001000301903552080		002 0007300003
&EXERCITVS	1EXERCITUM	CE001000301903514C00		002 0007300004
&EXERCITVS	2EXERCITUM	CE001000301903521A00	6	002 0007300005
&EXERCITVS	2EXERCITUM	CE001000301903521C00	5	002 0007300006
&HIEMO	HIEMARE	CE001000302003651071		002 0007400001
&ATQVE	1ATQUE	CE001000302103781000		002 0007500001
&ATQVE	2ATQUE	CE0010003021037820		002 0007500002
&	INVETERASCERE	CE0010003022038		002 0007600001
&IN	IN	CE001000302303970300		002 0007700001
&IN	IN	CE001000302303970600		002 0007700002
&GALLIA	NGALLIA	CE001000302404011A00		002 0007800001
&GALLIA	NGALLIA	CE001000302404011F00		002 0007800002
&MOLESTE	MOLESTE	CE001000302504160000		002 0007900001
&FERO	FEREBANT	CE001000302604256L12		002 0008000001
&PARTIM	PARTIM	CE001000302704360000		002 0008100001
&QVI	1QUI	CE001000302804446A	4	002 0008200001
&QVI	1QUI	CE001000302804446J	4	002 0008200002
&QVI	2QUI	CE0010003028044660		002 0008200003
&QVI	3QUI	CE0010003028044670		002 0008200004
&QVI	4QUI	CE001000302804460000		002 0008200005
&QVIS	1QUI	CE001000302804447A	4	002 0008200006

&QVIS	1QUI	CE0010003028044 47J	4	002	0008200007
&QVIS	2QUI	CE0010003028044 48A00	4	002	0008200008
&QVIS	2QUI	CE0010003028044 48J00	4	002	0008200009
&MOBILITAS	MOBILITATE	CE0010003029045 13F00		002	0008300001
&MOBILITO	MOBILITATE	CE0010003029045 51K21		002	0008300002
&ET	1ET	CE0010003030046 60000		002	0008400001
&ET	2ET	CE0010003030046 81000		002	0008400002
&LEVITAS	1LEVITATE	CE0010003031047 13F00		002	0008500001
&LEVITAS	2LEVITATE	CE0010003031047 13F00		002	0008500002
&ANIMVS	ANIMI	CE0010003032048 12D00		002	0008600001
&ANIMVS	ANIMI	CE0010003032048 12J00		002	0008600002
&NOVVS	NOVIS	CE0010003033049 21N00	1	002	0008700001
&NOVVS	NOVIS	CE0010003033049 21000	1	002	0008700002
&IMPERIVM	IMPERIIS	CE0010003034050 12N00		002	0008800001
&IMPERIVM	IMPERIIS	CE0010003034050 12000		002	0008800002
&STVDEO	STUDEBANT	CE0010003035051 52L12		0002	0008900001
&AB	AB	CE0010004001052 70600		002	0009000001
&NONNVLLVS	NONNULLIS	CE0010004002053 48N00	1	002	0009100001
&NONNVLLVS	NONNULLIS	CE0010004002053 48000	1	002	0009100002
&ETIAM	ETIAM	CE0010004003054 60000		002	0009200001
&QVI	1QUOD	CE0010004004055 46A	6	002	0009300001
&QVI	1QUOD	CE0010004004055 46C	6	002	0009300002
&QVIS	1QUOD	CE0010004004055 47A	6	002	0009300003
&QVIS	1QUOD	CE0010004004055 47C	6	002	0009300004
&QVIS	2QUOD	CE0010004004055 48A00	6	002	0009300005
&QVIS	2QUOD	CE0010004004055 48C00	6	002	0009300006
&QVOD	1QUOD	CE0010004004055 81000		002	0009300007
&QVOD	2QUOD	CE0010004004055 820		002	0009300008
&IN	IN	CE0010004005056 70300		002	0009400001
&IN	IN	CE0010004005056 70600		002	0009400002
&GALLIA	NGALLIA	CE0010004006057 11A00		002	0009500001
&GALLIA	NGALLIA	CE0010004006057 11F00		002	0009500002
&AB	A	CE0010004007058 70600		002	0009600001
&AH	A	CE0010004007058 90000		002	0009600002
&POTENS	POTENTIORIBUS	CE0010004008059 2EN00	1	002	0009700001
&POTENS	POTENTIORIBUS	CE0010004008059 2E000	1	002	0009700002
&ATQVE	1ATQUE	CE0010004009060 81000		002	0009800001
&ATQVE	2ATQUE	CE0010004009060 820		002	0009800002
&IS	IIS	CE0010004010061 45N00	1	002	0009900001
&IS	IIS	CE0010004010061 45000	1	002	0009900002
&QVI	1QUI	CE0010004011062 46A	4	002	0010000001
&QVI	1QUI	CE0010004011062 46J	4	002	0010000002
&QVI	2QUI	CE0010004011062 660		002	0010000003
&QVI	3QUI	CE0010004011062 670		002	0010000004
&QVI	4QUI	CE0010004011062 60000		002	0010000005
&QVIS	1QUI	CE0010004011062 47A	4	002	0010000006
&QVIS	1QUI	CE0010004011062 47J	4	002	0010000007
&QVIS	2QUI	CE0010004011062 48A00	4	002	0010000008
&QVIS	2QUI	CE0010004011062 48J00	4	002	0010000009
&AD	AD	CE0010004012063 70300		002	0010100001
&CONDVCO	CONDUCENDOS	CE0010004013064 5CL50	4	002	0010200001
&HOMO	HOMINES	CE0010004014065 13J00		002	0010300001
&HOMO	HOMINES	CE0010004014065 13L00		002	0010300002
&FACVLTAS	FACULTATES	CE0010004015066 13J00		002	0010400001
&FACVLTAS	FACULTATES	CE0010004015066 13L00		002	0010400002
&HABEO	HABEBANT	CE0010004016067 52L12		002	0010500001
&VVLGO	1VULGO	CE0010004017068 51A11		002	0010600001
&VVLGO	2VULGO	CE0010004017068 60000		002	0010600002
&VVLGVS	VULGO	CE0010004017068 12E00		002	0010600003
&VVLGVS	VULGO	CE0010004017068 12F00		002	0010600004
&REGNO	REGNA	CE0010004018069 51B21		002	0010700001
&REGNVM	REGNA	CE0010004018069 12J00		002	0010700002
&REGNVM	REGNA	CE0010004018069 12L00		002	0010700003

&OCCVPO	2OCCUPABANTUR	CE00100040190705AL12		002 0010800001
&QVI	1QUI	CE001000402007146A	4	002 0010900001
&QVI	1QUI	CE001000402007146J	4	002 0010900002
&QVI	2QUI	CE0010004020071660		002 0010900003
&QVI	3QUI	CE0010004020071670		002 0010900004
&QVI	4QUI	CE001000402007160000		002 0010900005
&QVIS	1QUI	CE001000402007147A	4	002 0010900006
&QVIS	1QUI	CE001000402007147J	4	002 0010900007
&QVIS	2QUI	CE001000402007148A00	4	002 0010900008
&QVIS	2QUI	CE001000402007148J00	4	002 0010900009
&PARVM	2MINUS	CE00100040210726&000		002 0011000001
&PARVVS	2MINUS	CE00100040210722AA00	6	002 0011000002
&PARVVS	2MINUS	CE00100040210722AB00	6	002 0011000003
&PARVVS	2MINUS	CE00100040210722AC00	6	002 0011000004
&FACILE	FACILE	CE001000402207360000		002 0011100001
&FACILIS	FACILE	CE001000402207324A00	6	002 0011100002
&FACILIS	FACILE	CE001000402207324C00	6	002 0011100003
&EO	1EAM	CE001000402307456A31		002 0011200001
&IS	EAM	CE001000402307445C00	2	002 0011200002
&RES	REM	CE001000402407515C00		002 0011300001
&IMPERIVM	IMPERIO	CE001000402507612E00		002 0011400001
&IMPERIVM	IMPERIO	CE001000402507612F00		002 0011400002
&NOSTER	NOSTRO	CE001000402607742E00	5	002 0011500001
&NOSTER	NOSTRO	CE001000402607742F00	5	002 0011500002
&CONSEQVOR	CONSEQUI	CE00100040270785L071		002 0011600001
&POSSVM	1POTERANT	CE00100040280795 6L12		K002 0011700001

ANNEXE 4 : FMOTLAT.AM

A	70600		AB	loin de
A	90000		AH	ah
AB	70600		AB	loin de
ABSUNT	56L11		ABSVM	ils sont éloignés de
AD	70300		AD	vers
ADDUCERETUR	5CC32		ADDVCO	il fut conduit
ADFEREBANTUR	5FL12		AFFERO	ils étaient apportés
ALIAM	48C00	2	ALIVS	autre
ANIMI	12D00		ANIMVS	de l'esprit
ANIMI	12J00		ANIMVS	les esprits
ANIMOS	12L00		ANIMVS	les coeurs
APPELLANTUR	5AL11		APPELLO	ils sont appelés
APPELLANTUR	5CL31		APPELLO	ils sont dirigés vers
AQUITANI			aquitani	les Aquitains
AQUITANIA			aquitania	l'Aquitaine
AQUITANIS	12000		aquitani	les Aquitains
ATQUE	81000		ATQVE	et
ATQUE	820		ATQVE	et
ATTINGIT	53C11		ATTINGO	il touche
AUT	81000		AVT	soit
BELGAE			belgae	les Belges
BELGARUM			belgae	des Belges
BELGAS			belgae	les Belges
BELGIS	11000		belgae	les Belges
BELLUM	12A00		BELLVM	la guerre
BELLUM	12C00		BELLVM	la guerre
BELLUM	21A00	6	BELLVS	joli
BELLUM	21C00	5	BELLVS	joli
CAESAR	13A00		CAESAR	César
CAESAR	13B00		CAESAR	César
CAPIT	55C11		CAPIO	il prend
CAUSA	11A00		CAVSA	la cause
CAUSA	11F00		CAVSA	la cause
CAUSAS	11L00		CAVSA	les causes
CELTAE			celtae	les Celtes
CERTIOR	2AA00	3	CERTVS	plus certain que
CITERIORE	2&F00	1	CITERIOR	plus rapproché
COMMEANT	51L11		COMMEO	ils circulent
CONDUCENDOS	5CL50	4	CONDVCO	**********
CONIURANDI	5AD50	5	CONIVRO	**********
CONIURANDI	5AJ50	4	CONIVRO	**********
CONIURANDI	51D60		CONIVRO	**********
CONIURARE	51071		CONIVRO	conspirer
CONSEQUI	5L071		CONSEQVOR	suivre
CONTENDUNT	53L11		CONTENDO	ils luttent
CONTINENTER	60000		CONTINENTER	continuellement
CONTINETUR	5BC11		CONTINEO	il est borné
CONTRA	60000		CONTRA	vis-à-vis
CONTRA	70300		CONTRA	contre
COTIDIANIS	21N00	1	COTIDIANVS	quotidiens
COTIDIANIS	21000	1	COTIDIANVS	quotidiens
CREBRI	21D00	5	CREBER	abondant
CREBRI	21J00	4	CREBER	abondants
CULTU	53090		COLO	**********
CULTU	14E00		CVLTVS	état de civilisation
CULTU	14F00		CVLTVS	état de civilisation
CUM	60000		CVM	en toutes circonstances
CUM	70600		CVM	avec
CUM	820		CVM	quand
DARE	51071		DO	donner
DE	70600		DE	de
DEINDE	60000		DEINDE	ensuite
DEMONSTRAVIMUS	51J14		DEMONSTRO	nous indiquons
DICTUM	5CA44	6	DICO	dit
DICTUM	5CC44	5	DICO	dit
DICTUM	53080		DICO	**********
DICTUM	12A00		DICTVM	la parole
DICTUM	12C00		DICTVM	la parole
DIFFERUNT	56L11		DIFFERO	ils diffèrent
DIUTIUS	6&000		DIV	longtemps
DIVIDIT	53C11		DIVIDO	il sépare
DIVISA	5CA44	2	DIVIDO	divisée

ANNEXE 4 : FMOTLAT.AM

DIVISA	5CF44	2	DIVIDO	divisée
DIVISA	5CJ44	6	DIVIDO	divisés
DIVISA	5CL44	6	DIVIDO	divisés
DIVISA	21A00	2	DIVISVS	divisée
DIVISA	21F00	2	DIVISVS	divisée
DIVISA	21J00	6	DIVISVS	divisés
DIVISA	21L00	6	DIVISVS	divisés
DIXERAMUS	53J15		DICO	nous avions dit
EA	60000		EA	par cet endroit
EA	45A00	2	IS	la
EA	45F00	2	IS	la
EA	45J00	6	IS	les
EA	45L00	6	IS	les
EAM	56A31		EO	que j'aille
EAM	45C00	2	IS	la
EFFEMINANDOS	5AL50	4	EFFEMINO	adoucis
EORUM	45M00	5	IS	d'eux
EOS	45L00	4	IS	les
ESSE	56071		EDO	manger
ESSE	E6071		SVM	être
ESSET	56C32		EDO	il mangeait
ESSET	E6C32		SVM	il était
EST	56C11		EDO	il mange
EST	E6C11		SVM	il est
ET	60000		ET	aussi
ET	81000		ET	et
ETIAM	60000		ETIAM	aussi
EUM	45C00	4	IS	le
EXERCITUM	5BA44	6	EXERCEO	exercé
EXERCITUM	5BC44	5	EXERCEO	exercé
EXERCITUM	52080		EXERCEO	**********
EXERCITUM	14C00		EXERCITVS	l'armée
EXERCITUM	21A00	6	EXERCITVS	tourmenté
EXERCITUM	21C00	5	EXERCITVS	tourmenté
EXERCITUS	5BA44	4	EXERCEO	exercé
EXERCITUS	14A00		EXERCITVS	l'armée
EXERCITUS	14D00		EXERCITVS	de l'armée
EXERCITUS	14J00		EXERCITVS	les armées
EXERCITUS	14L00		EXERCITVS	les armées
EXERCITUS	21A00	4	EXERCITVS	tourmenté
EXTREMIS	2JN00	1	EXTER	extrêmes
EXTREMIS	2JO00	1	EXTER	extrêmes
EXTREMIS	12N00		EXTREMA	les extrêmes
EXTREMIS	12O00		EXTREMA	les extrêmes
FACILE	60000		FACILE	facilement
FACILE	24A00	6	FACILIS	facile
FACILE	24C00	6	FACILIS	facile
FACULTATES	13J00		FACVLTAS	les possibilités
FACULTATES	13L00		FACVLTAS	les possibilités
FERE	60000		FERE	presque
FEREBANT	56L12		FERO	ils supportaient
FIEBAT	56C12		FIO	il portait
FINIBUS	13N00		FINIS	les frontières
FINIBUS	13O00		FINIS	les frontières
FLUMEN	13A00		FLVMEN	le fleuve
FLUMEN	13B00		FLVMEN	le fleuve
FLUMEN	13C00		FLVMEN	le fleuve
FLUMINE	13F00		FLVMEN	le fleuve
FLUMINIS	13D00		FLVMEN	du fleuve
FORTISSIMI	2MD00	5	FORTIS	le plus courageux
FORTISSIMI	2MJ00	4	FORTIS	les plus courageux
GALLI	12J00		GALLI	les Gaulois
GALLI	21D00	5	GALLVS	gaulois
GALLI	21J00	4	GALLVS	gaulois
GALLIA	11A00		GALLIA	la Gaule
GALLIA	11F00		GALLIA	la Gaule
GALLIAE	11D00		GALLIA	de la Gaule
GALLIAE	11E00		GALLIA	la Gaule
GALLIAE	11J00		GALLIA	les Gaules
GALLIS	12N00		GALLI	les Gaulois
GALLIS	12O00		GALLI	les Gaulois
GALLIS	21N00	1	GALLVS	gaulois

ANNEXE 4 : FMOTLAT.AM

GALLIS	21O00	1	GALLVS	gaulois
GALLOS	12L00		GALLI	les Gaulois
GALLOS	21L00	4	GALLVS	gaulois
GARUNNA	11A00		garunna	la Garonne
GERMANIS	11N00		GERMANA	les soeurs germaines
GERMANIS	11O00		GERMANA	les soeurs germaines
GERMANIS	21N00	1	GERMANVS	germains
GERMANIS	21O00	1	GERMANVS	germains
GERMANIS	12N00		GERMANVS	les Germains
GERMANIS	12O00		GERMANVS	les Germains
GERMANOS	21L00	4	GERMANVS	germains
GERMANOS	12L00		GERMANVS	les Germains
GERUNT	53L11		GERO	ils exécutent
HABEBANT	52L12		HABEO	ils avaient
HAS	45L00	2	HIC	celles-là
HELVETII			helvetii	les Helvètes
HELVETIIS			helvetii	les Helvètes
HI	45J00	4	HIC	ceux-là
HIEMARE	51071		HIEMO	hiverner
HISPANIAM	11C00		HISPANIA	l'Espagne
HOMINES	13J00		HOMO	les hommes
HOMINES	13L00		HOMO	les hommes
HORUM	45M00	5	HIC	de ceux-là
HUMANITATE	13F00		HVMANITAS	l'humanité
IIS	45N00	1	IS	les
IIS	45O00	1	IS	les
IMPERIIS	12N00		IMPERIVM	les maîtres
IMPERIIS	12O00		IMPERIVM	les maîtres
IMPERIO	12E00		IMPERIVM	le maître
IMPERIO	12F00		IMPERIVM	le maître
IMPORTANT	51L11		IMPORTO	ils introduisent
IN	70300		IN	dans
IN	70600		IN	dans
INCOLUNT	53L11		INCOLO	ils habitent
INFERIOREM	2AC00	3	INFERVS	inférieur
INITIUM	12A00		INITIVM	le début
INITIUM	12C00		INITIVM	le début
INSTITUTIS	5CN44	1	INSTITVO	**********
INSTITUTIS	5CO44	1	INSTITVO	**********
INSTITUTIS	12N00		INSTITVTVM	les coutumes
INSTITUTIS	12O00		INSTITVTVM	par les coutumes
INTER	70300		INTER	entre
INVETERASCERE			inveterascere	s'établir
IPSI	45E00	1	IPSE	lui-même
IPSI	45J00	4	IPSE	eux-mêmes
IPSORUM	45M00	5	IPSE	d'eux-mêmes
ITA	60000		ITA	ainsi
ITEM	60000		ITEM	pareillement
LABIENI	12D00		LABIENVS	le lieutenant de César
LABIENI	12J00		LABIENVS	les lieutenants de César
LEGIBUS	13N00		LEX	les lois
LEGIBUS	13O00		LEX	par les lois
LEVITATE	13F00		LEVITAS	la légèreté
LINGUA	11A00		LINGVA	la langue
LINGUA	11F00		LINGVA	par la langue
LITTERIS	11N00		LITTERA	les lettres
LITTERIS	11O00		LITTERA	les lettres
LONGISSIME	6-000		LONGE	le plus loin
MATRONA	11A00		MATRONA	la Marne
MATRONA	11F00		MATRONA	la Marne
MERCATORES	13J00		MERCATOR	les marchands
MERCATORES	13L00		MERCATOR	les marchands
MINIME	6-000		PARVM	le moins
MINUS	6&000		PARVM	moins
MINUS	2AA00	6	PARVVS	plus petit
MINUS	2AB00	6	PARVVS	plus petit
MINUS	2AC00	6	PARVVS	plus petit
MOBILITATE	13F00		MOBILITAS	la mobilité
MOBILITATE	51K21		MOBILITO	bougez !
MOLESTE	60000		MOLESTE	avec peine
MONTES	13J00		MONS	les montagnes
MONTES	13L00		MONS	les montagnes

ANNEXE 4 : FMOTLAT.AM

NE	60000		NE	ne pas
NE	670		NE	est-ce-que
NE	68000		NE	ne pas
NE	820		NE	pour que ne pas
NOLUERANT	56L15		NOLO	ils n'avaient pas voulu
NONNULLIS	48N00	1	NONNVLLVS	quelques-uns
NONNULLIS	48O00	1	NONNVLLVS	quelques-uns
NOSTER	42A00	4	NOSTER	notre
NOSTRA	42A00	2	NOSTER	notre
NOSTRA	42F00	2	NOSTER	notre
NOSTRA	42J00	6	NOSTER	nos
NOSTRA	42L00	6	NOSTER	nos
NOSTRO	42E00	5	NOSTER	notre
NOSTRO	42F00	5	NOSTER	notre
NOVIS	21N00	1	NOVVS	nouveaux
NOVIS	21O00	1	NOVVS	nouveaux
OBSIDES	13J00		OBSES	les otages
OBSIDES	13L00		OBSES	les otages
OBSIDES	52B11		OBSIDEO	tu assièges
OBSIDES	53B13		OBSIDO	tu assiègeras
OBTINERE	52071		OBTINEO	maintenir
OCCASUM	14C00		OCCASVS	la chute
OCCASUM	5CA44	6	OCCIDO	**********
OCCASUM	5CC44	5	OCCIDO	**********
OCCASUM	53080		OCCIDO	**********
OCCUPABANTUR	5AL12		OCCVPO	ils étaient occupés
OCEANI	12D00		OCEANVS	l'Océan
OCEANI	12J00		OCEANVS	les Océans
OCEANI	21D00	5	OCEANVS	aquatique
OCEANI	21J00	4	OCEANVS	aquatiques
OCEANO	12E00		OCEANVS	l'Océan
OCEANO	12F00		OCEANVS	l'Océan
OCEANO	21E00	5	OCEANVS	aquatique
OCEANO	21F00	5	OCEANVS	aquatique
OMNES	48J00	3	OMNIS	tous
OMNES	48L00	3	OMNIS	tous
OMNI	48E00	1	OMNIS	tout
OMNI	48F00	1	OMNIS	tout
OMNIS	48A00	3	OMNIS	tout
OMNIS	48D00	1	OMNIS	de tout
OMNIS	48L00	3	OMNIS	tous
OMNIUM	48M00	1	OMNIS	de tous
ORIENTEM	13C00		ORIENS	l'Est
ORIENTEM	5MC41	3	ORIOR	**********
ORIUNTUR	5ML11		ORIOR	ils naissent
PACATA	21A00	2	PACATVS	pacifique
PACATA	21F00	2	PACATVS	pacifique
PACATA	21J00	6	PACATVS	pacifiques
PACATA	21L00	6	PACATVS	pacifiques
PACATA	5AA44	2	PACO	soumis
PACATA	5AF44	2	PACO	soumis
PACATA	5AJ44	6	PACO	soumis
PACATA	5AL44	6	PACO	soumis
PARS	13A00		PARS	la partie
PARS	13B00		PARS	la partie
PARTEM	13C00		PARS	la partie
PARTES	13J00		PARS	les parties
PARTES	13L00		PARS	les parties
PARTIM	60000		PARTIM	en partie
PERTINENT	52L11		PERTINEO	ils aboutissent à
PERTINET	52C11		PERTINEO	il aboutit à
POPULI	12D00		POPVLVS	du peuple
POPULI	12J00		POPVLVS	les peuples
POPULUM	12C00		POPVLVS	le peuple
POTENTIORIBUS	2EN00	1	POTENS	plus puissants
POTENTIORIBUS	2EO00	1	POTENS	plus puissants
POTERANT	56L12		POSSVM	ils pouvaient
PRAECEDUNT	53L11		PRAECEDO	ils surpassent
PRIMUM	3N000		PRIMVM	premièrement
PRIMUM	3KA00	6	PRIMVS	le premier
PRIMUM	3KC00	5	PRIMVS	le premier
PROELIIS	12N00		PROELIVM	les combats

ANNEXE 4 : FMOTLAT.AM

PROELIIS	12O00		PROELIVM	les combats
PROHIBENT	52L11		PROHIBEO	ils interdisent
PROPTEREA	60000		PROPTEREA	à cause de cela
PROVINCIAE	11D00		PROVINCIA	de la province
PROVINCIAE	11E00		PROVINCIA	la province
PROVINCIAE	11J00		PROVINCIA	les provinces
PROXIMI	2-D00	5	PROPIOR	du plus proche
PROXIMI	2-J00	4	PROPIOR	les plus proches
PROXIMI	12D00		PROXIMVM	**********
PYRENAEOS	21L00	4	PYRENAEVS	pyrénéens
QUA	660		QVA	par où
QUA	670		QVA	par où ?
QUA	60000		QVA	**********
QUA	46F	2	QVI	laquelle
QUA	47F	2	QVIS	laquelle
QUA	48A00	2	QVIS	laquelle
QUA	48F00	2	QVIS	laquelle
QUA	48J00	6	QVIS	lesquelles
QUA	48L00	6	QVIS	lesquelles
QUAE	46A	2	QVI	laquelle
QUAE	46J	2	QVI	lesquelles
QUAE	46J	6	QVI	lesquelles
QUAE	46L	6	QVI	lesquelles
QUAE	47A	2	QVIS	laquelle
QUAE	47J	2	QVIS	lesquelles
QUAE	47J	6	QVIS	lesquelles
QUAE	47L	6	QVIS	lesquelles
QUAE	48A00	2	QVIS	laquelle
QUAE	48J00	2	QVIS	lesquelles
QUAE	48J00	6	QVIS	lesquelles
QUAE	48L00	6	QVIS	lesquelles
QUAM	660		QVAM	**********
QUAM	670		QVAM	**********
QUAM	46C	2	QVI	laquelle
QUAM	47C	2	QVIS	laquelle
QUAM	48C00	2	QVIS	laquelle
QUARUM	46M	2	QVI	desquelles
QUARUM	47M	2	QVIS	desquelles
QUARUM	48M00	2	QVIS	desquelles
QUE	81000		QVE	et
QUI	46A	4	QVI	qui
QUI	46J	4	QVI	qui
QUI	660		QVI	**********
QUI	670		QVI	**********
QUI	60000		QVI	**********
QUI	47A	4	QVIS	qui
QUI	47J	4	QVIS	qui
QUI	48A00	4	QVIS	quelqu'un
QUI	48J00	4	QVIS	quelqu'un
QUIBUS	46N	1	QVI	lesquels
QUIBUS	46O	1	QVI	lesquels
QUIBUS	47N	1	QVIS	quels
QUIBUS	47O	1	QVIS	quels
QUIBUS	48N00	1	QVIS	lesquels
QUIBUS	48O00	1	QVIS	lesquels
QUOD	46A	6	QVI	qui
QUOD	46C	6	QVI	qui
QUOD	47A	6	QVIS	lequel
QUOD	47C	6	QVIS	lequel
QUOD	48A00	6	QVIS	lequel
QUOD	48C00	6	QVIS	lequel
QUOD	81000		QVOD	parceque
QUOD	820		QVOD	parceque
QUOQUE	46F	5	QVISQVE	chacun
QUOQUE	48F00	5	QVISQVE	chacun
QUOQUE	60000		QVOQVE	aussi
REGNA	51B21		REGNO	règne !
REGNA	12J00		REGNVM	les royaumes
REGNA	12L00		REGNVM	les royaumes
RELIQUOS	12L00		RELIQVI	les restes
RELIQUOS	21L00	4	RELIQVVS	restant
REM	15C00		RES	la chose

ANNEXE 4 : FMOTLAT.AM

RHENI			rhenum	le Rhin
RHENUM			rhenum	le Rhin
RHODANO			rhodanus	le Rhône
ROMANI	12J00		ROMANI	les Romains
ROMANI	21D00	5	ROMANVS	romain
ROMANI	21J00	4	ROMANVS	romains
ROMANUM	21A00	6	ROMANVS	romain
ROMANUM	21C00	5	ROMANVS	romain
RUMORES	13J00		RVMOR	les rumeurs
RUMORES	13L00		RVMOR	les rumeurs
SAEPE	60000		SAEPE	souvent
SAEPE	13F00		SAEPES	la clôture
SE	43C00		SVI	eux-mêmes
SE	43F00		SVI	eux-mêmes
SEPTENTRIONEM	13C00		SEPTENTRIO	le Nord
SEPTENTRIONES	13J00		SEPTENTRIO	les contrées du Nord
SEPTENTRIONES	13L00		SEPTENTRIO	les contrées du Nord
SEQUANA	11A00		sequana	la Seine
SEQUANIS			sequani	les Séquanes
SOLEM	13C00		SOL	le soleil
SOLIS	13D00		SOL	du soleil
SOLIS	12N00		SOLVM	les soleils
SOLIS	12O00		SOLVM	les soleils
SOLIS	48N00	1	SOLVS	**********
SOLIS	48O00	1	SOLVS	**********
SOLLICITARENTUR	5AL32		SOLLICITO	ils sollicitaient
SPECTANT	51L11		SPECTO	ils regardent
SPECTAT	51C11		SPECTO	il regarde
STUDEBANT	52L12		STVDEO	ils rêvaient
SUIS	44N00	1	SVI	**********
SUIS	44O00	1	SVI	**********
SUIS	53B11		SVO	tu couds
SUIS	13D00		SVS	du porc
SUIS	12N00		SVVM	leur
SUIS	12O00		SVVM	leur
SUNT	56L11		SVM	ils sont
SUNT	E6L11		SVM	ils sont
SUPRA	60000		SVPRA	précédemment
SUPRA	70300		SVPRA	avant
TERTIAM	32C00	2	TERTIVS	troisième
TRANS	70300		TRANS	au-delà
TRES	31J00	3	TRES	trois
TRES	31L00	3	TRES	trois
UNA	60000		VNA	ensemble
UNA	31A00	2	VNVS	un
UNA	31F00	2	VNVS	un
UNA	31J00	6	VNVS	un
UNA	31L00	6	VNVS	un
UNAM	31C00	2	VNVS	un
UT	660		VT	**********
UT	670		VT	**********
UT	60000		VT	**********
UT	820		VT	**********
UTI	660		VT	**********
UTI	670		VT	**********
UTI	60000		VT	**********
UTI	820		VT	**********
UTI	5L071		VTOR	**********
VERERENTUR	5KL32		VEREOR	ils craignaient
VERGIT	53C11		VERGO	il tend vers
VERSARI	5A071		VERSO	faire tourner
VERSARI	5J071		VERSOR	faire tourner
VIRTUTE	13F00		VIRTVS	la vertu
VULGO	51A11		VVLGO	je divulgue
VULGO	60000		VVLGO	communément
VULGO	12E00		VVLGVS	la foule
VULGO	12F00		VVLGVS	la foule

ANNEXE 5 : INDEX.AM

ANNEXE 5 : INDEX.AM

ANNEXE 5 : INDEX.AM

```
Program CONVERT_LASLA_2;

uses ('$U b:globald2 ') GLOBALDECL2, CRT;

(' specifications : a partir d'un fichier fourni par le LASLA, CONVERT_LASLA_2  ')
('                  met a jour le fichier contenant les analyses morphologiques ')
('                  et les traductions fran\aises des mots latins 'b:fmotlat.am'')
('                  ainsi que le fichier index 'b:index.am' et cree le fichier  ')
('                  contenant le texte 'b:'.fct' et celui le d(crivant 'b:'.dsc'')

const
     tradnul = '**************************';
type
     ligne = str80;
var
     i, k, iocode, xcour, longreelle : integer;
     gotoend, ok, newfam, exist, insert, stop, nofamml : boolean;
     ch : char;
     stranalmorphlasla, stranalmorphlam : string[10];
     namelasla : string[12];
     nameflasla, namefct, namefdescr : str14;
     motlatinlasla, motdejaecrit, motaecrire : str20;
     flasla : text;
     ligneflasla : ligne;
     firstamml, lamml, newlamml, lammlcour : list_motlat_analmorph;
     lam, newlam, lamcour : list_anal_morph_lat;

label FIN, MAJFAMML;

(*************************************************************************)

procedure COPIE_FAM_LAM;

(' sp(cifications : copie les donn(es du fichier des analyses morphologiques ')
('                  dans une liste chai^n(e plus facilement manipulable.      ')

begin
      lammlcour := nil;
      lamcour := nil;
      motdejaecrit := '';
      while ( not eof(famml) ) do
      begin
           read (famml, lignefamml);
           if (lignefamml.motlat <> motdejaecrit)
           then begin
                     new (newlamml);
                     newlamml^.motlat := lignefamml.motlat;
                     new (newlam);
                     with newlam^.analmorphlat do
                     begin catgram := lignefamml.catgram;
                           sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                           cas_pers_nbr := lignefamml.cas_pers_nbr;
                           mode := lignefamml.mode;
                           temps := lignefamml.temps;
                           fonction := lignefamml.fonction;
                           emplois := lignefamml.emplois;
                           genre := lignefamml.genre;
                           codesubord[1] := lignefamml.codesubord[1];
                           codesubord[2] := lignefamml.codesubord[2];
                           lemme := lignefamml.lemme;
                           tradf := lignefamml.tradf
                     end;
                     newlam^.analmorphlatsuivant := nil;
                     newlamml^.analmorphmotlat := newlam;
                     newlamml^.motlatsuivant := nil;
```

```
                    if (lammlcour = nil)
                    then begin
                              lammlcour := newlamml;
                              firstamml := newlamml
                         end
                    else begin
                              lammlcour^.motlatsuivant := newlamml;
                              lammlcour := lammlcour^.motlatsuivant
                         end;
                    lamcour := newlam;
                    motdejaecrit := lignefamml.motlat
               end
          else begin
                    new (newlam);
                    with newlam^.analmorphlat do
                    begin catgram := lignefamml.catgram;
                         sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                         cas_pers_nbr := lignefamml.cas_pers_nbr;
                         mode := lignefamml.mode;
                         temps := lignefamml.temps;
                         fonction := lignefamml.fonction;
                         emplois := lignefamml.emplois;
                         genre := lignefamml.genre;
                         codesubord[1] := lignefamml.codesubord[1];
                         codesubord[2] := lignefamml.codesubord[2];
                         lemme := lignefamml.lemme;
                         tradf := lignefamml.tradf
                    end;
                    newlam^.analmorphlatsuivant := nil;
                    lamcour^.analmorphlatsuivant := newlam;
                    lamcour := lamcour^.analmorphlatsuivant
               end
     end;
     close (famml)
end;

(*************************************************************************)

procedure COPIE_LAM_FAM;

(* sp{cifications : recopie les donn{es de la liste chai^n{e dans le fichier  *)
(*                  des analyses morphologiques et cr{e le fichier 'index'   *)
(*                  associ{.                                                 *)

begin
     assign (findexfamml, 'b:index.am');
     rewrite (findexfamml);
     rewrite(famml);
     lamml := firstamml;
     while (lamml <> nil) do
     begin
          ligneindex.motlatin := lamml^.motlat;
          ligneindex.pos := filepos(famml);
          write (findexfamml, ligneindex);
          lam := lamml^.analmorphmotlat;
          while (lam <> nil) do
          begin
               with lignefamml do
               begin
                    motlat := lamml^.motlat;
                    catgram := lam^.analmorphlat.catgram;
                    sscatgram_dgr_vx := lam^.analmorphlat.sscatgram_dgr_vx;
                    cas_pers_nbr := lam^.analmorphlat.cas_pers_nbr;
                    mode := lam^.analmorphlat.mode;
```

```
                    temps := lam^.analmorphlat.temps;
                    fonction := lam^.analmorphlat.fonction;
                    emplois := lam^.analmorphlat.emplois;
                    genre := lam^.analmorphlat.genre;
                    codesubord[1] := lam^.analmorphlat.codesubord[1];
                    codesubord[2] := lam^.analmorphlat.codesubord[2];
                    lemme := lam^.analmorphlat.lemme;
                    tradf := lam^.analmorphlat.tradf
               end;
               write (famml, lignefamml);
               lam := lam^.analmorphlatsuivant
          end;
          lamml := lamml^.motlatsuivant
     end;
     close (famml);
     close (findexfamml)
end;

(*************************************************************************)

procedure REMPLI_LAM;

(* sp{cifications : mise @ jour de la liste chai^n{e contenant des analyses   *)
(*                  morphologiques @ partir des donn{es concernant un mot latin*)

begin
     lamml := firstamml;
     lammlcour := nil;
     stranalmorphlasla := copy (ligneflasla, 54, 10);
     insert := false;
     while ( (lamml <> nil) and (insert = false) ) do
     begin
          if (lamml^.motlat = motlatinlasla)
          then begin
                    exist := false;
                    lam := lamml^.analmorphmotlat;
                    while ( (exist = false) and (lam <> nil) ) do
                    begin
                         stranalmorphlam := concat ( lam^.analmorphlat.catgram, lam^.analmorphlat.sscatgram_dgr_vx,
                                                     lam^.analmorphlat.cas_pers_nbr, lam^.analmorphlat.mode,
                                                     lam^.analmorphlat.temps, lam^.analmorphlat.fonction,
                                                     lam^.analmorphlat.emplois, lam^.analmorphlat.genre,
                                                     lam^.analmorphlat.codesubord[1], lam^.analmorphlat.codesubord[2]);
                         if (stranalmorphlasla <> stranalmorphlam)
                         then lam := lam^.analmorphlatsuivant
                         else exist := true
                    end;
                    if (exist = false)
                    then begin (* inserer l'analyse morphologique *)
                              lam := lamml^.analmorphmotlat;
                              while (lam^.analmorphlatsuivant <> nil) do lam := lam^.analmorphlatsuivant;
                              new (newlam);
                              with newlam^.analmorphlat do
                              begin
                                   catgram := ligneflasla[54];
                                   sscatgram_dgr_vx := ligneflasla[55];
                                   cas_pers_nbr := ligneflasla[56];
                                   mode := ligneflasla[57];
                                   temps := ligneflasla[58];
                                   fonction := ligneflasla[59];
                                   emplois := ligneflasla[60];
                                   genre := ligneflasla[61];
                                   codesubord[1] := ligneflasla[62];
                                   codesubord[2] := ligneflasla[63];
                                   lemme := copy (ligneflasla, 2, 16);
```

```
                                tradf := tradnul
                            end;
                            newlam^.analmorphlatsuivant := nil;
                            lam^.analmorphlatsuivant := newlam
                        end;
                    insert := true
                end;
            if (lamml^.motlat < motlatinlasla)
            then begin
                    lammlcour := lamml;
                    lamml := lamml^.motlatsuivant
                end;
            if (lamml^.motlat > motlatinlasla)
            then begin
                    new (newlamml);
                    newlamml^.motlat := motlatinlasla;
                    new (newlam);
                    with newlam^.analmorphlat do
                    begin
                        catgram := ligneflasla[54];
                        sscatgram_dgr_vx := ligneflasla[55];
                        cas_pers_nbr := ligneflasla[56];
                        mode := ligneflasla[57];
                        temps := ligneflasla[58];
                        fonction := ligneflasla[59];
                        emplois := ligneflasla[60];
                        genre := ligneflasla[61];
                        codesubord[1] := ligneflasla[62];
                        codesubord[2] := ligneflasla[63];
                        lemme := copy (ligneflasla, 2, 16);
                        tradf := tradnul
                    end;
                    newlam^.analmorphlatsuivant := nil;
                    newlamml^.motlatsuivant := lamml;
                    newlamml^.analmorphmotlat := newlam;
                    if ( lammlcour = nil )
                    then firstamml := newlamml
                    else begin
                            lammlcour^.motlatsuivant := newlamml;
                            lammlcour := lammlcour^.motlatsuivant
                        end;
                    insert := true
                end
    end;
    if ( (lamml = nil) and (insert = false) )
    then begin (* ajout a la fin *)
            new (newlamml);
            newlamml^.motlat := motlatinlasla;
            new (newlam);
            with newlam^.analmorphlat do
            begin
                catgram := ligneflasla[54];
                sscatgram_dgr_vx := ligneflasla[55];
                cas_pers_nbr := ligneflasla[56];
                mode := ligneflasla[57];
                temps := ligneflasla[58];
                fonction := ligneflasla[59];
                emplois := ligneflasla[60];
                genre := ligneflasla[61];
                codesubord[1] := ligneflasla[62];
                codesubord[2] := ligneflasla[63];
                lemme := copy (ligneflasla, 2, 16);
                tradf := tradnul
            end;
            newlam^.analmorphlatsuivant := nil;
```

```
                newlamml^.motlatsuivant := nil;
                newlamml^.analmorphmotlat := newlam;
                if ( lammlcour = nil )
                then firstamml := newlamml
                else lammlcour^.motlatsuivant := newlamml
            end
end;

(*****************************************************************************)


BEGIN
      clrscr;
      gotoend := false;
      iocode := 0;
(*$I-*)
      repeat
            writeln ('Nom du fichier LASLA @ convertir ou RETURN : B :  __________');
            gotoxy (51, wherey-1);
            readln (namelasla);
            if namelasla = ''
            then gotoend := true
            else begin
                      nameflasla := concat ('b:',namelasla);
                      assign (flasla, nameflasla);
                      reset (flasla);
                      iocode := ioresult;
                      if iocode <> 0
                      then writeln (nameflasla,' n''existe pas !')
                 end
      until ( (iocode = 0) or (gotoend = true) );
(*$I+*)
      if (gotoend = true) then goto FIN;
      namefct := copy (nameflasla, 1, 10);
      namefct := concat (namefct, '.fct');
(*$I-*)
      assign (fconttxt, namefct);
      reset (fconttxt);
      iocode := ioresult;
      nofamml := false;
      if iocode = 0
      then begin
                ok := false;
                writeln (namefct, ' existe d(j@ ! ');
                writeln ('     1- ''overwrite'' de tous les fichiers ');
                writeln ('     2- mise @ jour seulement du fichier des analyses morphologiques');
                writeln ('     3- mise @ jour seulement des fichiers texte et descriptions ');
                writeln ('     4- fin ');
                write ('     votre choix (1, 2, 3 ou 4) : ');
                while (ok = false) do
                begin
                     if keypressed
                     then begin
                               ch := readkey;
                               case ch of
                                    '1','2','3','4' : begin
                                                           ok := true;
                                                           writeln (ch)
                                                      end;
                                    else begin
                                              sound (220);
                                              delay (200);
                                              nosound
                                         end
                               end
```

```
                    end
                end;
                if (ch = '4') then begin
                                        close (flasla);
                                        close (fconttxt);
                                        goto FIN
                                    end;
                if (ch = '3') then begin
                                        close (fconttxt);
                                        nofamml := true
                                    end;
                if (ch = '2') then begin
                                        close (fconttxt);
                                        goto MAJFAMML
                                    end;
                if (ch = '1') then close (fconttxt);
            end;
(*$I+*)

(* creations du fichier contenant le texte : fichconttxt (*.fct) *)
(*       et du fichier decrivant ce texte : fdescrtxt   (*.dsc) *)

      rewrite (fconttxt);
      namefdescr := copy (nameflasla, 1, 10);
      namefdescr := concat (namefdescr, '.dsc');
(*$I-*)
      assign (fdescrtxt, namefdescr);
      reset (fdescrtxt);
      iocode := ioresult;
      if (iocode = 0) then close (fdescrtxt);
(*$I+*)
      rewrite (fdescrtxt);
      lignefdt.typedescr := txtdescr;
      writeln ('Titre de ce texte latin (maximum 30 caract}res) : ');
      write ('______________________________');
      gotoxy (1, wherey);
      readln (lignefdt.titre_txt);
      with lignefdt do
      begin
            etat_prep_txt := enprep;
            res_trt_aut := nontrtaut;
            comment_txt := '';
            nomfichcontxt := namefct
      end;
      write (fdescrtxt, lignefdt);
      i := 0;            (* i = nombre de phrases dans le texte *)
      motdejaecrit := '';
      xcour := 0;
      while (not eof(flasla)) do
      begin
            read (flasla, ligneflasla);
            motlatinlasla := copy (ligneflasla, 19, 20);
            if (motlatinlasla <> motdejaecrit)
            then begin
                      stop := false;
                      k := 1;
                      repeat
                            if (motlatinlasla[k] = ' ')
                            then stop := true
                            else k := k + 1
                      until ((k > 20) or (stop = true) );
                      longreelle := k - 1;
                      if (xcour = 0) then lignefconttxt := '';
                      motaecrire := copy (motlatinlasla, 1, longreelle);
                      if ((80 - xcour) >= (longreelle + 1))
```

```
                then begin
                        if ( (ligneflasla[66] <> 'K') and (ligneflasla[66] <> 'S')
                        and  (ligneflasla[66] <> '2') )
                        then begin
                                lignefconttxt := concat (lignefconttxt, motaecrire, ' ');
                                xcour := xcour + longreelle + 1
                            end
                        else begin
                                lignefconttxt := concat (lignefconttxt, motaecrire, '.');
                                xcour := xcour + longreelle + 1;
                                if (xcour < 80)
                                then for k := (xcour + 1) to 80 do lignefconttxt[k] := ' ';
                                writeln (fconttxt, lignefconttxt);
                                xcour := 0;
                                i := i + 1;
                                lignefdt.typedescr := phrdescr;
                                with lignefdt do
                                begin
                                     numero_phrase := i;
                                     etat_prep_phrase := e1;
                                     res_trt_aut_phrase := nontrt;
                                     comment_phrase := '';
                                     nomfichanal := ''
                                end;
                                write (fdescrtxt, lignefdt)
                            end
                    end
                else begin
                        if (xcour < 80)
                        then for k := (xcour + 1) to 80 do lignefconttxt[k] := ' ';
                        writeln (fconttxt, lignefconttxt);
                        if ( (ligneflasla[66] <> 'K') and (ligneflasla[66] <> 'S')
                        and  (ligneflasla[66] <> '2') )
                        then begin
                                lignefconttxt := concat (motaecrire, ' ');
                                xcour := longreelle + 1
                            end
                        else begin
                                lignefconttxt := concat (motaecrire, '.');
                                xcour := longreelle + 1;
                                if (xcour < 80)
                                then for k := (xcour + 1) to 80 do lignefconttxt[k] := ' ';
                                writeln (fconttxt, lignefconttxt);
                                xcour := 0;
                                i := i + 1;
                                lignefdt.typedescr := phrdescr;
                                with lignefdt do
                                begin
                                     numero_phrase := i;
                                     etat_prep_phrase := e1;
                                     res_trt_aut_phrase := nontrt;
                                     comment_phrase := '';
                                     nomfichanal := ''
                                end;
                                write (fdescrtxt, lignefdt)
                            end
                    end;
                motdejaecrit := motlatinlasla
            end;
        readln (flasla)
end;
close (fconttxt);
close (fdescrtxt);
writeln ('fin creations fconttxt et fdescrtxt');
if (nofamml = true) then begin
```

```
                                close(flasla);
                                goto FIN
                            end;

(* mise a jour du fichier contenant les analyses morphologiques des mots latins *)
MAJFAMML :
(*$I-*)
     assign (famml, 'b:fmotlat.am');
     reset (famml);
     iocode := ioresult;
     if (iocode <> 0) then newfam := true
                      else newfam := false;
(*$I+*)
     IF (newfam = true)
     THEN begin
               close (flasla);
               reset (flasla);
               if (not eof(flasla))
               then begin
                         read (flasla, ligneflasla);
                         new (newlamml);
                         motlatinlasla := copy (ligneflasla, 19, 20);
                         newlamml^.motlat := motlatinlasla;
                         new (newlam);
                         with newlam^.analmorphlat do
                         begin
                              catgram := ligneflasla[54];
                              sscatgram_dgr_vx := ligneflasla[55];
                              cas_pers_nbr := ligneflasla[56];
                              mode := ligneflasla[57];
                              temps := ligneflasla[58];
                              fonction := ligneflasla[59];
                              emplois := ligneflasla[60];
                              genre := ligneflasla[61];
                              codesubord[1] := ligneflasla[62];
                              codesubord[2] := ligneflasla[63];
                              lemme := copy (ligneflasla, 2, 16);
                              tradf := tradnul
                         end;
                         newlam^.analmorphlatsuivant := nil;
                         newlamml^.analmorphmotlat := newlam;
                         newlamml^.motlatsuivant := nil;
                         firstamml := newlamml;
                         lammlcour := newlamml;
                         readln (flasla)
                    end;
               while (not eof(flasla)) do
               begin (* on va continuer a remplir la liste selon l'ordre alphabetique *)
                    read (flasla, ligneflasla);
                    motlatinlasla := copy (ligneflasla, 19, 20);
                    REMPLI_LAM;
                    readln (flasla)
               end;
               close (flasla);
             (* apres avoir creer et mis a jour alphabetiquement la liste chainee, *)
             (* il faut la recopier dans le fichier famml.                         *)
               COPIE_LAM_FAM
          end
     ELSE begin (* famml existe deja : il va falloir recopier ce fichier dans la liste chainee, *)
               (* manipuler, mettre a jour cette liste et puis la recopier dans le fichier     *)
               COPIE_FAM_LAM;
               close (flasla);
               reset (flasla);
               while ( not eof(flasla) ) do
               begin
```

```
                    read (flasla, ligneflasla);
                    motlatinlasla := copy (ligneflasla, 19, 20);
                    REMPLI_LAM;
                    readln (flasla)
                end;
                close (flasla);
                COPIE_LAM_FAM
            end;
        writeln ('fin maj fammm');

FIN :
        writeln (' conversion termin(e !')
END.
```

```
Program GESTAM;

uses ('$U b:globald2 ') GLOBALDECL2, CRT;

(* sp{cifications : GESTAM permet de mettre @ jour d'une mani}re interactive *)
(*                  le fichier 'b:fmotlat.am' en ce qui concerne les A.M.,les*)
(*                  traductions et les lemmes associ{s aux mots latins.      *)

const
     motb = '                        ';

var
     ok, stop : boolean;
     ch : char;
     motdejaecrit : str20;
     bigmot : string[48];
     firstamml, newlamml, lammlcour : list_motlat_analmorph;
     newlam, lamcour : list_anal_morph_lat;
     analmorphol : string[10];

Begin
     clrscr;
     assign (famml, 'b:fmotlat.am');
     reset (famml);
     stop := false;
     lammlcour := nil;
     lamcour := nil;
     motdejaecrit := '';
     while ( (not eof(famml)) and (stop = false) ) do
     begin
          read (famml, lignefamml);
          if (lignefamml.motlat <> motdejaecrit)
          then begin
                    new (newlamml);
                    newlamml^.motlat := lignefamml.motlat;
                    new (newlam);
                    write (lignefamml.motlat);
                    write (' ', lignefamml.catgram);
                    write (' ', lignefamml.sscatgram_dgr_vx);
                    write (' ', lignefamml.cas_pers_nbr);
                    write (' ', lignefamml.mode);
                    write (' ', lignefamml.temps);
                    write (' ', lignefamml.fonction);
                    write (' ', lignefamml.emplois);
                    write (' ', lignefamml.genre);
                    write (' ', lignefamml.codesubord[1]);
                    writeln (' ', lignefamml.codesubord[2]);
                    write ('Voulez-vous changer cette analyse morphologique ? ''O''-''N'' : _');
                    gotoxy (wherex - 1, wherey);
                    ok := false;
                    while (ok = false) do
                    begin
                         if keypressed
                         then begin
                                   ch := readkey;
                                   case ch of
                                        'O', 'o' : begin
                                                        writeln (ch);
                                                        ok := true;
                                                        write ('Analyse morphologique : _________ ');
                                                        gotoxy (25, wherey);
                                                        readln (analmorphol);
                                                        bigmot := concat (analmorphol, motb, '            ');
                                                        with newlam^.analmorphlat do
                                                        begin
```

```
                                        catgram := bigmot[1];
                                        sscatgram_dgr_vx := bigmot[2];
                                        cas_pers_nbr := bigmot[3];
                                        mode := bigmot[4];
                                        temps := bigmot[5];
                                        fonction := bigmot[6];
                                        emplois := bigmot[7];
                                        genre := bigmot[8];
                                        codesubord[1] := bigmot[9];
                                        codesubord[2] := bigmot[10]
                                    end
                                end;
                            'N', 'n' : begin
                                        writeln (ch);
                                        ok := true;
                                        with newlam^.analmorphlat do
                                        begin
                                        catgram := lignefamml.catgram;
                                        sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                                        cas_pers_nbr := lignefamml.cas_pers_nbr;
                                        mode := lignefamml.mode;
                                        temps := lignefamml.temps;
                                        fonction := lignefamml.fonction;
                                        emplois := lignefamml.emplois;
                                        genre := lignefamml.genre;
                                        codesubord[1] := lignefamml.codesubord[1];
                                        codesubord[2] := lignefamml.codesubord[2]
                                    end
                                end;
                            else begin
                                        sound (220);
                                        delay (200);
                                        nosound
                                end
                    end
            end
end;
write ('Lemme associé : ', lignefamml.lemme);
write (' Changement ? ''O'', ''N'' : ');
ok := false;
while (ok = false) do
begin
     if keypressed
     then begin
               ch := readkey;
               case ch of
                    'O', 'o' : begin
                                        writeln (ch);
                                        ok := true;
                                        write ('Lemme @ associer : _____ ');
                                        gotoxy (20, wherey);
                                        readln (newlam^.analmorphlat.lemme);
                                        bigmot := concat (newlam^.analmorphlat.lemme, motb, '       ');
                                        newlam^.analmorphlat.lemme := copy (bigmot, 1, 16)
                                end;
                            'N', 'n' : begin
                                        writeln (ch);
                                        ok := true;
                                        newlam^.analmorphlat.lemme := lignefamml.lemme
                                end;
                            else begin
                                        sound (220);
                                        delay (200);
                                        nosound
                                end
```

```
                end
            end
end;
write ('Traduction associ{e : ', lignefamml.tradf);
write (' Changement ? ''O'', ''N'' : ');
ok := false;
while (ok = false) do
begin
     if keypressed
     then begin
               ch := readkey;
               case ch of
                    'O', 'o' : begin
                                    writeln (ch);
                                    ok := true;
                                    write ('Traduction @ associer : _____ ');
                                    gotoxy (25, wherey);
                                    readln (newlam^.analmorphlat.tradf);
                                    bigmot := concat (newlam^.analmorphlat.tradf, motb);
                                    newlam^.analmorphlat.tradf := copy (bigmot, 1, 24)
                               end;
                    'N', 'n' : begin
                                    writeln (ch);
                                    ok := true;
                                    newlam^.analmorphlat.tradf := lignefamml.tradf;
                               end;
                          else begin
                                    sound (220);
                                    delay (200);
                                    nosound
                               end
               end
          end
end;
newlam^.analmorphlatsuivant := nil;
newlamml^.analmorphmotlat := newlam;
newlamml^.motlatsuivant := nil;
if (lammlcour = nil)
then begin
          lammlcour := newlamml;
          firstamml := newlamml
     end
else begin
          lammlcour^.motlatsuivant := newlamml;
          lammlcour := lammlcour^.motlatsuivant
     end;
lamcour := newlam;
motdejaecrit := lignefamml.motlat;
writeln ('Tapez une touche pour continuer ou ''ESC''');
writeln;
ch := readkey;
if (ch = #27) then begin
                        stop := true;
                        while (not eof(famml)) do
                        begin
                             read (famml, lignefamml);
                             if (lignefamml.motlat <> motdejaecrit)
                             then begin
                                       new (newlamml);
                                       newlamml^.motlat := lignefamml.motlat;
                                       new (newlam);
                                       with newlam^.analmorphlat do
                                       begin
                                            catgram := lignefamml.catgram;
                                            sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
```

```
                                        cas_pers_nbr := lignefamml.cas_pers_nbr;
                                        mode := lignefamml.mode;
                                        temps := lignefamml.temps;
                                        fonction := lignefamml.fonction;
                                        emplois := lignefamml.emplois;
                                        genre := lignefamml.genre;
                                        codesubord[1] := lignefamml.codesubord[1];
                                        codesubord[2] := lignefamml.codesubord[2];
                                        lemme := lignefamml.lemme;
                                        tradf := lignefamml.tradf
                                    end;
                                    newlam^.analmorphlatsuivant := nil;
                                    newlamml^.analmorphmotlat := newlam;
                                    newlamml^.motlatsuivant := nil;
                                    if (lammlcour = nil)
                                    then begin
                                              lammlcour := newlamml;
                                              firstamml := newlamml
                                         end
                                    else begin
                                              lammlcour^.motlatsuivant := newlamml;
                                              lammlcour := lammlcour^.motlatsuivant
                                         end;
                                    lamcour := newlam;
                                    motdejaecrit := lignefamml.motlat
                                end
                           else begin
                                    new (newlam);
                                    with newlam^.analmorphlat do
                                    begin
                                         catgram := lignefamml.catgram;
                                         sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                                         cas_pers_nbr := lignefamml.cas_pers_nbr;
                                         mode := lignefamml.mode;
                                         temps := lignefamml.temps;
                                         fonction := lignefamml.fonction;
                                         emplois := lignefamml.emplois;
                                         genre := lignefamml.genre;
                                         codesubord[1] := lignefamml.codesubord[1];
                                         codesubord[2] := lignefamml.codesubord[2];
                                         lemme := lignefamml.lemme;
                                         tradf := lignefamml.tradf
                                    end;
                                    newlam^.analmorphlatsuivant := nil;
                                    lamcour^.analmorphlatsuivant := newlam;
                                    lamcour := lamcour^.analmorphlatsuivant
                                end
                           end
                      end
                 end
            end
       else begin
                 write (lignefamml.motlat);
                 write (' ', lignefamml.catgram);
                 write (' ', lignefamml.sscatgram_dgr_vx);
                 write (' ', lignefamml.cas_pers_nbr);
                 write (' ', lignefamml.mode);
                 write (' ', lignefamml.temps);
                 write (' ', lignefamml.fonction);
                 write (' ', lignefamml.emplois);
                 write (' ', lignefamml.genre);
                 write (' ', lignefamml.codesubord[1]);
                 writeln (' ', lignefamml.codesubord[2]);
                 new (newlam);
                 write ('Voulez-vous changer cette analyse morphologique ? ''O''-''N'' : _');
                 gotoxy (wherex - 1, wherey);
```

```
ok := false;
while (ok = false) do
begin
      if keypressed
      then begin
                ch := readkey;
                case ch of
                     '0', 'o' : begin
                                      writeln (ch);
                                      ok := true;
                                      write ('Analyse morphologique : _________ ');
                                      gotoxy (25, wherey);
                                      readln (analmorphol);
                                      bigmot := concat (analmorphol, motb, '              ');
                                      with newlam^.analmorphlat do
                                      begin
                                            catgram := bigmot[1];
                                            sscatgram_dgr_vx := bigmot[2];
                                            cas_pers_nbr := bigmot[3];
                                            mode := bigmot[4];
                                            temps := bigmot[5];
                                            fonction := bigmot[6];
                                            emplois := bigmot[7];
                                            genre := bigmot[8];
                                            codesubord[1] := bigmot[9];
                                            codesubord[2] := bigmot[10]
                                      end
                                end;
                           'N', 'n' : begin
                                      writeln (ch);
                                      ok := true;
                                      with newlam^.analmorphlat do
                                      begin
                                            catgram := lignefamml.catgram;
                                            sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                                            cas_pers_nbr := lignefamml.cas_pers_nbr;
                                            mode := lignefamml.mode;
                                            temps := lignefamml.temps;
                                            fonction := lignefamml.fonction;
                                            emplois := lignefamml.emplois;
                                            genre := lignefamml.genre;
                                            codesubord[1] := lignefamml.codesubord[1];
                                            codesubord[2] := lignefamml.codesubord[2]
                                      end
                                end;
                           else begin
                                      sound (220);
                                      delay (200);
                                      nosound
                                end
                end
          end
end;
write ('Lemme associ{ : ', lignefamml.lemme);
write (' Changement ? ''O'', ''N'' : ');
ok := false;
while (ok = false) do
begin
     if keypressed
     then begin
                ch := readkey;
                case ch of
                     '0', 'o' : begin
                                      writeln (ch);
                                      ok := true;
```

```
                                        write ('Lemme @ associer : _____________ ');
                                        gotoxy (20, wherey);
                                        readln (newlam^.analmorphlat.lemme);
                                        bigmot := concat (newlam^.analmorphlat.lemme,
                                                          motb, '        ');
                                        newlam^.analmorphlat.lemme := copy (bigmot, 1, 16)
                                    end;
                            'N', 'n' : begin
                                        writeln (ch);
                                        ok := true;
                                        newlam^.analmorphlat.lemme := lignefamml.lemme
                                    end;
                                else begin
                                        sound (220);
                                        delay (200);
                                        nosound
                                    end
                        end
                end
        end;
        write ('Traduction associ{e : ', lignefamml.tradf);
        write (' Changement ? ''O'', ''N'' : ');
        ok := false;
        while (ok = false) do
        begin
             if keypressed
             then begin
                        ch := readkey;
                        case ch of
                             'O', 'o' : begin
                                        writeln (ch);
                                        ok := true;
                                        write ('Traduction @ associer : _____________________ ');
                                        gotoxy (25, wherey);
                                        readln (newlam^.analmorphlat.tradf);
                                        bigmot := concat (newlam^.analmorphlat.tradf, motb);
                                        newlam^.analmorphlat.tradf := copy (bigmot, 1, 24)
                                    end;
                            'N', 'n' : begin
                                        writeln (ch);
                                        ok := true;
                                        newlam^.analmorphlat.tradf := lignefamml.tradf;
                                    end;
                                else begin
                                        sound (220);
                                        delay (200);
                                        nosound
                                    end
                        end
                end
        end;
        newlam^.analmorphlatsuivant := nil;
        lamcour^.analmorphlatsuivant := newlam;
        lamcour := lamcour^.analmorphlatsuivant;
        writeln ('Tapez une touche pour continuer ou ''ESC''');
        writeln;
        ch := readkey;
        if (ch = #27) then begin
                            stop := true;
                            while (not eof(famml)) do
                            begin
                                 read (famml, lignefamml);
                                 if (lignefamml.motlat <> motdejaecrit)
                                 then begin
                                           new (newlamml);
```

```
                                        newlamml^.motlat := lignefamml.motlat;
                                        new (newlam);
                                        with newlam^.analmorphlat do
                                        begin
                                              catgram := lignefamml.catgram;
                                              sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                                              cas_pers_nbr := lignefamml.cas_pers_nbr;
                                              mode := lignefamml.mode;
                                              temps := lignefamml.temps;
                                              fonction := lignefamml.fonction;
                                              emplois := lignefamml.emplois;
                                              genre := lignefamml.genre;
                                              codesubord[1] := lignefamml.codesubord[1];
                                              codesubord[2] := lignefamml.codesubord[2];
                                              lemme := lignefamml.lemme;
                                              tradf := lignefamml.tradf
                                        end;
                                        newlam^.analmorphlatsuivant := nil;
                                        newlamml^.analmorphmotlat := newlam;
                                        newlamml^.motlatsuivant := nil;
                                        if (lammlcour = nil)
                                        then begin
                                                  lammlcour := newlamml;
                                                  firstamml := newlamml
                                             end
                                        else begin
                                                  lammlcour^.motlatsuivant := newlamml;
                                                  lammlcour := lammlcour^.motlatsuivant
                                             end;
                                        lamcour := newlam;
                                        motdejaecrit := lignefamml.motlat
                                   end
                              else begin
                                        new (newlam);
                                        with newlam^.analmorphlat do
                                        begin
                                              catgram := lignefamml.catgram;
                                              sscatgram_dgr_vx := lignefamml.sscatgram_dgr_vx;
                                              cas_pers_nbr := lignefamml.cas_pers_nbr;
                                              mode := lignefamml.mode;
                                              temps := lignefamml.temps;
                                              fonction := lignefamml.fonction;
                                              emplois := lignefamml.emplois;
                                              genre := lignefamml.genre;
                                              codesubord[1] := lignefamml.codesubord[1];
                                              codesubord[2] := lignefamml.codesubord[2];
                                              lemme := lignefamml.lemme;
                                              tradf := lignefamml.tradf
                                        end;
                                        newlam^.analmorphlatsuivant := nil;
                                        lamcour^.analmorphlatsuivant := newlam;
                                        lamcour := lamcour^.analmorphlatsuivant
                                   end
                              end
                         end
                    end;
               close (famml);
               rewrite (famml);
               lammlcour := firstamml;
               while (lammlcour <> nil) do
               begin
                    lamcour := lammlcour^.analmorphmotlat;
                    while (lamcour <> nil) do
                    begin
```

```
                with lignefamml do
                begin
                    motlat := lammlcour^.motlat;
                    catgram := lamcour^.analmorphlat.catgram;
                    sscatgram_dgr_vx := lamcour^.analmorphlat.sscatgram_dgr_vx;
                    cas_pers_nbr := lamcour^.analmorphlat.cas_pers_nbr;
                    mode := lamcour^.analmorphlat.mode;
                    temps := lamcour^.analmorphlat.temps;
                    fonction := lamcour^.analmorphlat.fonction;
                    emplois := lamcour^.analmorphlat.emplois;
                    genre := lamcour^.analmorphlat.genre;
                    codesubord[1] := lamcour^.analmorphlat.codesubord[1];
                    codesubord[2] := lamcour^.analmorphlat.codesubord[2];
                    lemme := lamcour^.analmorphlat.lemme;
                    tradf := lamcour^.analmorphlat.tradf
                end;
                write (famml, lignefamml);
                lamcour := lamcour^.analmorphlatsuivant
            end;
            lammlcour := lammlcour^.motlatsuivant
        end;
        close (famml)
End.
```

```
Program TESTS;

uses ('$U b:globald2 ') GLOBALDECL2, CRT;

const
     colortxt = 14;    (' yellow ')
     colorfond = 0;    (' black  ')

var
     ch : char;
     out, stop : boolean;

label DEBUT;

(*****************************************************************************)

procedure TESTFCT;

var
   iocode  : integer;
   namefdt : str14;

label fin;

begin
     clrscr;
     iocode := 0;
('$I-')
     repeat
           writeln ('nom complet du fichier ''.fct'' @ visionner : _____');
           gotoxy (45, wherey-1);
           readln (namefdt);
           if namefdt = ''
           then goto fin
           else begin
                     assign (fconttxt, namefdt);
                     reset (fconttxt);
                     iocode := ioresult;
                     if iocode <> 0
                     then writeln (namefdt, ' n''existe pas !')
                end
      until (iocode = 0);
('$I+')
     stop := false;
     while ( (not eof (fconttxt)) and (stop = false) ) do
     begin
          readln (fconttxt, lignefconttxt);
          write (lignefconttxt);
          if (wherex = 1) then gotoxy (80, wherey - 1);
          repeat until keypressed;
          ch := readkey;
          if (ch = #27) then stop := true;
          writeln
     end;
     if (stop = false) then begin
                                writeln;
                                writeln;
                                write ('Tapez une touche pour retourner au menu !');
                                repeat until keypressed;
                                ch := readkey
                           end;
     close (fconttxt);
fin :
     writeln
end;
```

```
(*************************************************************************)

procedure TESTDSC;

var
    i, iocode : integer;
    namefdsc  : str14;

label FIN;

begin
     clrscr;
     iocode := 0;
(*$I-*)
     repeat
          writeln ('nom complet du fichier ''.dsc'' @ visionner : _____ ');
          gotoxy (45, wherey-1);
          readln (namefdsc);
          if namefdsc = '' then goto FIN
                           else begin
                                     assign (fdescrtxt, namefdsc);
                                     reset (fdescrtxt);
                                     iocode := ioresult;
                                     if (iocode <> 0 ) then writeln (namefdsc, ' n''existe pas !')
                                end
     until (iocode = 0);
(*$I+*)
     stop := false;
     while ( (not eof(fdescrtxt)) and (stop = false) ) do
     begin
          read (fdescrtxt, lignefdt);
          case lignefdt.typedescr of
              txtdescr : begin writeln ('description g{n{rale du texte :');
                               writeln (lignefdt.titre_txt);
                               case lignefdt.etat_prep_txt of
                                    pret : writeln ('pret');
                                    enprep : writeln ('en preparation')
                               end;
                               case lignefdt.res_trt_aut of
                                    nontrtaut : writeln ('non trait{ automatiquement');
                                    incoherence : writeln ('incoh{rence');
                                    ok : writeln ('ok')
                               end;
                               writeln (lignefdt.comment_txt);
                               writeln (lignefdt.nomfichcontxt);
                               writeln
                         end;
             phrdescr : begin writeln ('phrase', lignefdt.numero_phrase);
                               case lignefdt.etat_prep_phrase of
                                    e1 : writeln ('e1= analyse du LASLA de Li}ge');
                                    e2 : writeln ('e2= compl{ments manuels de l''analyse en cours');
                                    e3 : writeln ('e3= compl{ments manuels de l''analyse termin{s mais non v{rifi{s');
                                    e4 : writeln ('e4= compl{ments manuels de l''analyse incoh{rents');
                                    e5 : writeln ('e5= analyse coh{rente')
                               end;
                               case lignefdt.res_trt_aut_phrase of
                                    incoherente : writeln ('incoherente');
                                    coherente : writeln ('coherente');
                                    nontrt : writeln ('non trait{e')
                               end;
                               writeln (lignefdt.comment_phrase);
                               writeln (lignefdt.nomfichanal);
                               writeln
                         end;
```

```
            end;
            repeat until keypressed;
            ch := readkey;
            if (ch = #27) then stop := true
      end;
      if (stop = false) then begin
                                    writeln;
                                    writeln;
                                    write ('Tapez une touche pour retourner au menu !');
                                    repeat until keypressed;
                                    ch := readkey
                              end;
      close (fdescrtxt);
FIN :
      writeln
end;

(*************************************************************************)

procedure TESTFAMML;

begin
      clrscr;
      assign (famml, 'b:fmotlat.am');
      reset (famml);
      stop := false;
      while ( (not eof(famml)) and (stop = false) ) do
      begin
            read (famml, lignefamml);
            write (lignefamml.motlat);
            write (' ',lignefamml.catgram);
            write (lignefamml.sscatgram_dgr_vx);
            write (lignefamml.cas_pers_nbr);
            write (lignefamml.mode);
            write (lignefamml.temps);
            write (lignefamml.fonction);
            write (lignefamml.emplois);
            write (lignefamml.genre);
            write (lignefamml.codesubord[1]);
            write (lignefamml.codesubord[2]);
            write (' ', lignefamml.lemme);
            writeln (' ', lignefamml.tradf);
            repeat until keypressed;
            ch := readkey;
            if (ch = #27) then stop := true
      end;
      if (stop = false) then begin
                                    writeln;
                                    writeln;
                                    write ('Tapez une touche pour retourner au menu !');
                                    repeat until keypressed;
                                    ch := readkey
                              end;
      close (famml)
end;

(*************************************************************************)

procedure TESTINDEX;

begin
      clrscr;
      assign (findexfamml, 'b:index.am');
      reset (findexfamml);
```

```
      stop := false;
      while ( (not eof (findexfamml)) and (stop = false) ) do
      begin
            read (findexfamml, ligneindex);
            write (ligneindex.motlatin);
            gotoxy (25, wherey);
            writeln (ligneindex.pos);
            repeat until keypressed;
            ch := readkey;
            if (ch = #27) then stop := true
      end;
      if (stop = false) then begin
                                 writeln;
                                 writeln;
                                 write ('Tapez une touche pour retourner au menu !');
                                 repeat until keypressed;
                                 ch := readkey
                           end;
      close (findexfamml)
end;

(*****************************************************************************)

procedure TESTAN;

var
      iocode  : integer;
      namefna : str14;

label fin;

begin
      clrscr;
      iocode := 0;
(*$I-*)
      repeat
            writeln ('nom complet du fichier d''analyse @ visionner : _____');
            gotoxy (48, wherey - 1);
            readln (namefna);
            if (namefna ='')
            then  goto fin
            else begin
                       assign (fna, namefna);
                       reset (fna);
                       iocode := ioresult;
                       if iocode <> 0 then writeln (namefna, ' n''existe pas !')
                 end
      until (iocode = 0);
(*$I+*)
      stop := false;
      while ((not eof(fna)) and (stop = false)) do
      begin
            read (fna, fnaligne);
            write ('Noeud Indicatif ou Break : ');
            case fnaligne.noeud.ind_brk of
                 indicatif : writeln ('Indicatif');
                 break     : writeln ('Break')
            end;
            write ('Appartenance de ce noeud @ une analyse correcte de la phrase : ');
            case fnaligne.noeud.appart_analyse_correct of
                 true  : writeln ('Oui');
                 false : writeln ('Non');
                 else    writeln ('?')
            end;
            writeln ('Commentaire du syst)me associ{ @ ce noeud : ');
```

```
            writeln (fnaligne.noeud.comment_syst);
            writeln ('Commentaire sp{cifique associ{ @ ce noeud : ');
            writeln (fnaligne.noeud.comment_specif);
            write ('Acceptation de l''affichage du commentaire : ');
            case fnaligne.noeud.accept_comment of
                true  : writeln ('Oui');
                false : writeln ('Non')
            end;
            writeln ('Type de ce noeud : ');
            case fnaligne.noeud.typenoeud of
                n1 : begin
                         writeln ('1 : choix d''un mot associ{ @ une fonction');
                     end;
                n2 : begin
                         writeln ('2 : analyse morphologique d''un mot');
                     end;
                n3 : begin
                         writeln ('3 : association de mots @ un symbole syntaxique');
                     end;
                n4 : begin
                         writeln ('4 : traductions des mots de la phrase')
                     end
            end;
            write ('Existence d''un noeud fils de ce noeud : ');
            case fnaligne.noeudfils of
                true  : writeln ('Oui');
                false : writeln ('Non')
            end;
            write ('Existence d''un noeud frere de ce noeud : ');
            case fnaligne.noeudfrere of
                true  : writeln ('Oui');
                false : writeln ('Non')
            end;
            repeat until keypressed;
            ch := readkey;
            if (ch = #27) then stop := true;
            writeln;
        end;
        close (fna);
fin :
        writeln
end;

(************************ programme principal TESTS ************************)

BEGIN
DEBUT :
    textcolor (colortxt);
    textbackground (colorfond);
    window (1, 1, 80, 25);
    clrscr;
    gotoxy (25, 3);
    write ('TESTS DE CONTENUS DE FICHIERS :');
    gotoxy (25, 4);
    write ('-----------------------------');
    gotoxy (5, 9);
    writeln(' 1- CONTENU D''UN FICHIER ''TEXTE'' (''b:*.fct'')');
    writeln;
    writeln ('     2- CONTENU D''UN FICHIER DE DESCRIPTION D''UN TEXTE (''b:*.dsc'')');
    writeln;
    writeln ('     3- CONTENU DU FICHIER DES ANALYSES MORPHOLOGIQUES (''b:fmotlat.am'')');
    writeln;
    writeln ('     4- CONTENU DU FICHIER D''INDEX DES ANALYSES MORPHOLOGIQUES (''b:index.am'')');
    writeln;
    writeln ('     5- CONTENU D''UN FICHIER D''ANALYSE D''UNE PHRASE LATINE');
```

```
     writeln;
     writeln ('     6- FIN');
     gotoxy (18, 21);
     write ('TAPEZ LE NUMERO CORRESPONDANT A VOTRE CHOIX : _');
     gotoxy (64, 21);
     out := false;
     while (out = false) do
     begin
          if keypressed
          then begin
                    ch := readkey;
                    case ch of
                         '1' : begin
                                    write (ch);
                                    TESTFCT;
                                    goto DEBUT
                               end;
                         '2' : begin
                                    write (ch);
                                    TESTDSC;
                                    goto DEBUT
                               end;
                         '3' : begin
                                    write (ch);
                                    TESTFAMML;
                                    goto DEBUT
                               end;
                         '4' : begin
                                    write (ch);
                                    TESTINDEX;
                                    goto DEBUT
                               end;
                         '5' : begin
                                    write (ch);
                                    TESTAN;
                                    goto DEBUT
                               end;
                         '6' : begin
                                    write (ch);
                                    out := true
                               end;
                        else  begin
                                    sound (220);
                                    delay (200);
                                    nosound
                               end
                    end
               end
     end
END.
```

```
Unit GLOBALDECL2 ;
Interface
   type
        traitement = (trtv, trtgns, trtgncod, trtgncplt, traduct, trtnul);
        str2 = string[2];
        str16 = string[16];
        str20 = string[20];
        str24 = string[24];
        str30 = string[30];
        trttxt = (nontrtaut, incoherence, ok);
        preptxt = (pret,enprep);
        trtphrase= (incoherente, coherente, nontrt);
        prepphrase = (e1, e2, e3, e4, e5);
(* e1= analyse fournie par le LASLA de Liege                              *)
(* e2= complements manuels de l'analyse en cours                         *)
(* e3= complements manuels de l'analyse termines mais non verifies       *)
(* e4= complements manuels de l'analyse incoherents                      *)
(* e5= analyse coherente, prete                                          *)

        anal_morph_lat = record
                           catgram : char;
                           sscatgram_dgr_vx : char;
                           cas_pers_nbr : char;
                           mode : char;
                           temps : char;
                           fonction : char;
                           emplois : char;
                           genre : char;
                           codesubord : str2;
                           lemme : str16;
                           tradf : str24
                         end;
        list_anal_morph_lat = ^lanalmorphlat;
        lanalmorphlat = record
                          analmorphlat : anal_morph_lat;
                          analmorphlatsuivant : list_anal_morph_lat
                        end;
        list_motlat_analmorph = ^lmotlatanalmorph;
        lmotlatanalmorph = record
                             motlat : str20;
                             analmorphmotlat : list_anal_morph_lat;
                             motlatsuivant : list_motlat_analmorph
                           end;


        listtradfranc = ^ltradfranc;
        ltradfranc = record
                       motlatin : str20;
                       tradfranc : str24;
                       suivant_tf : listtradfranc
                     end;

        typenoeudaa = (n1, n2, n3, n4);
(* n1= choix d'un mot correspondant a une fonction       *)
(* n2= analyse morphologique d'un mot                    *)
(* n3= association d'un ensemble de mots a un symbole categoriel synt.*)
(* n4= traduction admise de tous les mots de la phrase                *)
        typecategorie = (subst, adj, num, adjpron, v, adv, prep, conj, interj);
(* types de cat{gories possibles pour d{crire morphologiquement un mot latin *)
        typeadjpron = (pers, pos, refl, posrefl, dem, relat, inter, ind);
(* types possibles d'un adjectif pronom latin : PERSsonnel, POSsessif, REFL{chi, *)
(* POSsessif REFL{chi, DEMonstratif, RELATif, INTERrogatif ou IND{fini.          *)
        cas = (nom,voc,acc,gen,datif,abl,loc,indecl);
        nombre = (sing, pluriel,nombrenul);
        genre = (commun, fem, mascfem, masc, mascneutre, neutre,genrenul);
```

```
        mode = (indic,imp,subj,part,adjverb,gerondif,inf,supinum,supinu, modenul);
        temps = (pres,impft,futspl,prft,plusqueprft,futant,fuisse, tempsnul);
        degre = (positif, comparatif, superlatif);
        personne = (p1, p2, p3, pnul);
        str80 = string[80];
        type_ib = (indicatif, break);
        arrbool85 = array [1..85] of boolean;
        noeudaa = record
                    ind_brk : type_ib;
                    appart_analyse_correct : boolean;
                    comment_syst : str80;
                    comment_specif : str80;
                    accept_comment : boolean;
                    case typenoeud : typenoeudaa of
                        n1 : ( fonction : (sujet, verbe, cod, cplt, somethingelse);
                               mot1 : str20);
                        n2 : ( lemme : str16;
                               case categorie : typecategorie of
                                   subst : ( declsubst : (d1,d2,d3,d4,d5,d6,d7);
                                             cassubst : cas;
                                             nbrsubst : nombre;
                                             genresubst : genre );
                                   adj : ( classeadj : (a1,a2,a3,a4,a5,a6,a7);
                                           degreadj : degre;
                                           casadj : cas ;
                                           nbradj : nombre;
                                           genreadj : genre);
                                   num : ( degrenum : degre;
                                           catnum : (card,ord,distr,mult,advord,advmult);
                                           nbrnum : nombre;
                                           casnum : cas;
                                           genrenum : genre );
                                   adjpron : ( catadjpron : typeadjpron;
                                               nbradjpron : nombre;
                                               casadjpron : cas;
                                               genreadjpron : genre;
(* si pronom relatif ou interrogatif *)           modesubadjpron : mode;
(* si pronom relatif ou interrogatif *)           tempssubadjpron : temps );
                                   v : ( conjv : (c1,c2,c3,c4,c5,c6);
                                         voixv : (actif,passif,deponent,semideponent);
                                         fonctionv : (princ, subord, fonctionnul);
                                         genrev : genre;
                                         tempsv : temps;
                                         case modev : mode of
                                             indic, imp, subj,
                                             inf, supinum, supinu : (persv : personne;
                                                                     nbr1v : nombre);
(* si inf, supinum ou supinu : persv = nbr1v = 0 *)
(* si supinum ou supinu : tempsv = 0 et si inf : tempsv = 1 *)
(* si adjverb ou gerondif : tempsv = 0 et si part : tempsv = 4 ou 1 *)
                                             part, adjverb, gerondif : (casv : cas;
                                                                        nbr2v : nombre) );
                                   adv : (catadv : (rel, int, neg, intneg, comp, superl, catnul);
(* si interrogatif ou relatif *)                 modesubadv : mode;
(* si interrogatif ou relatif *)                 tempssubadv : temps );
                                   prep : ( typemecum : boolean;
                                            casprep : (accusatif,genitif,ablatif) );
                                   conj : (catconj : (coord, sub);
(* si subordonn(e *)                             modesubconj : mode;
(* si subordonn(e *)                             tempssubconj : temps );
                                   interj : ( ) );
                        n3 : ( symbcatsynt : (gnsujet, gncod, gnabl, whatelse);
                               ensmots3 : arrbool85);
          (* 85 = limite maximum de mots dans une phrase (?)   *)
          (* un boolean correspond @ un mot : accept( ou pas   *)
```

```
                    n4 : ( )
                end;
        listnoeudaa = ^lnoeudaa;
        lnoeudaa = record
                    noeud : noeudaa;
                    noeudfils : listnoeudaa;
                    noeudfrere : listnoeudaa;
                    noeudpere : listnoeudaa
                   end;

        fich_cont_txt = text;   (* une ligne = 80 char, longueur r{elle des *)
                                (* mots, point, blanc s{parant les mots et  *)
                                (* nouvelle phrase = nouvelle ligne.        *)
                                (* fichier "fixe" .                         *)

        analmorphmotlat =  record
                             motlat : str20;
                             catgram : char;
                             sscatgram_dgr_vx : char;
                             cas_pers_nbr : char;
                             mode : char;
                             temps : char;
                             fonction : char;
                             emplois : char;
                             genre : char;
                             codesubord : str2;
                             lemme : str16;
                             tradf : str24
                           end;
        fich_analmorph_motlat = file of analmorphmotlat;
                             (* fichier modifiable et valable pour tout texte        *)


        fnoeudaa = record
                    noeud : noeudaa;
                    noeudfils : boolean;
                    noeudfrere : boolean
                   end;
        fich_arbreanalyse = file of fnoeudaa;

        str14 = string[14];
        tdescr = (txtdescr, phrdescr);
        descr_txt = record
                    case typedescr : tdescr of
(* seek (0) *)           txtdescr : (titre_txt : str30;
                                     etat_prep_txt : preptxt;
                                     res_trt_aut : trttxt;
                                     comment_txt : str80;
                                     nomfichcontxt : str14);
(* seek (nphr) *)        phrdescr : (numero_phrase : integer;
                                     etat_prep_phrase : prepphrase;
                                     res_trt_aut_phrase : trtphrase;
                                     comment_phrase : str80;
                                     nomfichanal : str14)
                    end;
        fich_descr_txt = file of descr_txt;     (* fichier contenant des informations g{n{rales modifiables @  *)
                                                (* propos du texte entier et plus particuli}rement ses phrases.*)

        list_descr = ^descr1;
        descr1 = record
                   contenu : descr_txt;
                   suivant : list_descr
                 end;

        indexam = record
```

```
                motlatin : str20;
                pos : integer
              end;
      indexfamml = file of indexam;

      lcontphr = ^mcontphr;
      mcontphr = record
                  mot : str20;
                  libre : boolean;
                  suivant : lcontphr
                 end;


  var
      fdescrtxt : fich_descr_txt;      (* pour chaque texte : 'b:*.dsc' *)
      lignefdt : descr_txt;
      fconttxt : fich_cont_txt;        (* pour chaque texte : 'b:*.fct' *)
      lignefconttxt : str80;
      fna : fich_arbreanalyse;         (* pour chaque phrase : 'b:nomfile.nphrase' *)
      fnaligne : fnoeudaa;
      famml : fich_analmorph_motlat;   (* unique : 'b:fmotlat.am' *)
      lignefamml : analmorphmotlat;
      ligneindex : indexam;
      findexfamml : indexfamml;        (* unique : 'b:index.am' *)

    (* donc pour traiter un texte, il faut le fichier texte, le fichier de description, le fichier *)
    (* d'analyses et les fichiers g{n{raux : 'analyses morphologiques + lexique' et son index.    *)

implementation

begin
end.
```

```
Unit LP_UNIT;

Interface
uses (*$U b:globald2 *) GLOBALDECL2, CRT, DOS;

   type str60 = string[60];

   procedure VISUTXTPHR ( var namefct : str14; nphrase : integer;
                          var title : str60; miny : integer;
                          maxy : integer; var result : boolean );

   procedure COPY_FDSC_LDSC (var namefdt : str14; var firstld : list_descr);

   procedure COPY_LDSC_FDSC (var namefdt : str14; var firstld : list_descr);

   procedure TRSF_PHR_L ( var namefct : str14; nphrase : integer; var fmcp : lcontphr);

   procedure FIND_INDEX ( var mot : str20; var where : integer);

   procedure COPY_FA_LA ( var nomfichanal : str14; var firstlna : listnoeudaa);

   procedure COPY_LA_FA ( var nomfichanal : str14; var firstlna : listnoeudaa);

   function VALID_TXT (var nametxt : str14) : boolean;

   procedure LIST_TXT_LAT;

   procedure VISU_TXT_LAT (var namefcttxt : str14);

   procedure LIST_RES_TRT_AUT (var namefdt : str14);

   procedure VISU_ETAT_PREP_TXT_LAT (var namefdt : str14);



Implementation


procedure VISUTXTPHR;

(* sp{cifications : Si 'nphrase' = 0                                    *)
(*                  Alors VISUTXTPHR affichera @ l'{cran entre les lignes   *)
(*                        'miny' et 'maxy' d'abord 'title', puis toutes les *)
(*                        phrases contenues dans 'namefct'. Si le texte est *)
(*                        trop long que pour apparai^tre en un {cran, la    *)
(*                        pression d'une touche permettra d'en voir la suite.*)
(*                        'result' vaudra alors true.                       *)
(*                  Sinon VISUTXTPHR affichera @ l'{cran entre les lignes   *)
(*                        'miny' et 'maxy' d'abord 'title', puis la phrase  *)
(*                        portant le num{ro 'nphrase' dans le texte 'namefct'*)
(*                        Si la phrase est trop longue que pour apparai^tre en*)
(*                        un {cran, la pression d'une touche permettra d'en *)
(*                        voir la suite.  'result' vaudra true si la phrase *)
(*                        existe dans le texte, sinon il vaudra 'false'.    *)

var
    i : integer;
    finphrase : boolean;
    ch : char;

begin
     clrscr;
```

```
      assign (fconttxt, namefct);
      reset (fconttxt);
      result := true;
      if (nphrase = 0)
      then begin
                gotoxy (20, miny);
                writeln (title);
                writeln;
                while ( not eof(fconttxt) ) do
                begin
                     readln (fconttxt, lignefconttxt);
                     if (wherey > maxy)
                     then begin
                               gotoxy (35, maxy + 2);
                               write ('Tapez une touche pour voir la page suivante');
                               repeat until keypressed;
                               ch := readkey;
                               clrscr;
                               gotoxy (20, miny);
                               writeln (title);
                               writeln
                          end;
                     write (lignefconttxt);
                     if (wherex = 1) then gotoxy (80, wherey - 1);
                     writeln
                end
          end
      else begin
                i := 0;
                while ( (not eof(fconttxt)) and (i < nphrase - 1) ) do
                begin
                     readln (fconttxt, lignefconttxt);
                     if (lignefconttxt[length(lignefconttxt)] = '.') then i := i + 1
                end;
                if ( (eof(fconttxt)) and (i <= nphrase - 1) )
                then result := false
                else begin
                          gotoxy (20, miny);
                          writeln (title);
                          writeln;
                          finphrase := false;
                          while ( (not eof(fconttxt)) and (finphrase = false) ) do
                          begin
                               readln (fconttxt, lignefconttxt);
                               if (wherey > maxy)
                               then begin
                                         gotoxy (35, maxy + 2);
                                         write ('Tapez une touche pour voir la page suivante');
                                         repeat until keypressed;
                                         ch := readkey;
                                         clrscr;
                                         gotoxy (20, miny);
                                         writeln (title);
                                         writeln
                                    end;
                               write (lignefconttxt);
                               if (wherex = 1) then gotoxy (80, wherey - 1);
                               writeln;
                               if (lignefconttxt[length(lignefconttxt)] = '.') then finphrase := true
                          end
                     end

          end;
      close (fconttxt)
end;
```

```
(*************************************************************************)

procedure COPY_FDSC_LDSC;

(* specifications : copie le fichier d(crivant un texte latin dans une liste *)
(*                  chai^n(e plus facilement manipulable.                    *)

var
    ldcour, newld : list_descr;

begin
      assign (fdescrtxt, namefdt);
      reset (fdescrtxt);
      ldcour := nil;
      seek (fdescrtxt, 0);
      while ( not eof(fdescrtxt) ) do
      begin
            read (fdescrtxt, lignefdt);
            new (newld);
            if (lignefdt.typedescr = txtdescr )
            then with newld^.contenu do
                 begin
                       typedescr := lignefdt.typedescr;
                       titre_txt := lignefdt.titre_txt;
                       etat_prep_txt := lignefdt.etat_prep_txt;
                       res_trt_aut := lignefdt.res_trt_aut;
                       comment_txt := lignefdt.comment_txt;
                       nomfichcontxt := lignefdt.nomfichcontxt
                 end
            else with newld^.contenu do
                 begin
                       typedescr := lignefdt.typedescr;
                       numero_phrase := lignefdt.numero_phrase;
                       etat_prep_phrase := lignefdt.etat_prep_phrase;
                       res_trt_aut_phrase := lignefdt.res_trt_aut_phrase;
                       comment_phrase := lignefdt.comment_phrase;
                       nomfichanal := lignefdt.nomfichanal
                 end;
            newld^.suivant := nil;
            if ( ldcour = nil )
            then begin
                       ldcour := newld;
                       firstld := newld
                 end
            else begin
                       ldcour^.suivant := newld;
                       ldcour := ldcour^.suivant
                 end
      end;
      close (fdescrtxt)
end;


(*************************************************************************)

procedure COPY_LDSC_FDSC;

(* sp(cifications : recopie dans le fichier associ( les donn(es de        *)
(*                  description d'un texte contenues dans la liste chai^n(e. *)

var
    ldcour : list_descr;

begin
```

```
      assign (fdescrtxt, namefdt);
      rewrite (fdescrtxt);
      ldcour := firstld;
      while (ldcour <> nil ) do
      begin
            if (ldcour^.contenu.typedescr = txtdescr)
            then begin
                      with lignefdt do
                      begin
                            typedescr := ldcour^.contenu.typedescr;
                            titre_txt := ldcour^.contenu.titre_txt;
                            etat_prep_txt := ldcour^.contenu.etat_prep_txt;
                            res_trt_aut := ldcour^.contenu.res_trt_aut;
                            comment_txt := ldcour^.contenu.comment_txt;
                            nomfichcontxt := ldcour^.contenu.nomfichcontxt
                      end
                end
            else begin
                      with lignefdt do
                      begin
                            typedescr := ldcour^.contenu.typedescr;
                            numero_phrase := ldcour^.contenu.numero_phrase;
                            etat_prep_phrase := ldcour^.contenu.etat_prep_phrase;
                            res_trt_aut_phrase := ldcour^.contenu.res_trt_aut_phrase;
                            comment_phrase := ldcour^.contenu.comment_phrase;
                            nomfichanal := ldcour^.contenu.nomfichanal
                      end
                end;
            write (fdescrtxt, lignefdt);
            ldcour := ldcour^.suivant
      end;
      close (fdescrtxt)
end;

(*************************************************************************)

procedure TRSF_PHR_L;

(* sp{cifications : TRSF_PHR_L re\oit 'nphrase', le num{ro d'une phrase      *)
(*                  appartenant au texte latin m{moris{ dans le fichier      *)
(*                  portant le nom 'namefct', et recopie cette phrase dans   *)
(*                  une liste chai^n{e plus facilement manipulable.          *)
(*                  fmcp contiendra un pointeur pointant vers le premier     *)
(*                  maillon de cette liste chai^n{e.                         *)

const
       motb = '                    ';
var
    j, k, min : integer;
    finphrase, finligne : boolean;
    mcpcour, newmcp : lcontphr;
    bigmot : string[40];

begin
      assign (fconttxt, namefct);
      reset (fconttxt);
      j := 0;
      while ( (not eof(fconttxt)) and (j < nphrase - 1) ) do
      begin
            readln (fconttxt, lignefconttxt);
            if ( lignefconttxt[length(lignefconttxt)] = '.' ) then j := j + 1;
      end;
      finphrase := false;
      mcpcour := nil;
      while ( (not eof(fconttxt)) and (finphrase = false) ) do
```

```
      begin
            readln (fconttxt, lignefconttxt);
            k := 0;
            finligne := false;
            min := 1;
            while ( (k < 80) and (finligne = false) ) do
            begin
                  repeat k := k + 1 until ( (k = 80) or (lignefconttxt[k] = ' ') or (lignefconttxt[k] = '.') );
                  new (newmcp);
                  if ( (k = 80) and (lignefconttxt[k] <> ' ') and (lignefconttxt[k] <> '.') )
                  then begin
                              newmcp^.mot := copy (lignefconttxt, min, k - min + 1);
                              bigmot := concat (newmcp^.mot, motb);
                              newmcp^.mot := copy (bigmot, 1, 20)
                        end;
                  if ( (k = 80) and ( (lignefconttxt[k] = ' ') or (lignefconttxt[k] = '.') ) )
                  then begin
                              newmcp^.mot := copy (lignefconttxt, min, k - min);
                              bigmot := concat (newmcp^.mot, motb);
                              newmcp^.mot := copy (bigmot, 1, 20)
                        end;
                  if (k < 80)
                  then begin
                              newmcp^.mot := copy (lignefconttxt, min, k - min);
                              bigmot := concat (newmcp^.mot, motb);
                              newmcp^.mot := copy (bigmot, 1, 20);
                              if ( (lignefconttxt[k] = '.') or (lignefconttxt[k+1] = ' ') )
                              then finligne := true
                              else min := k + 1
                        end;
                  newmcp^.libre := true;
                  newmcp^.suivant := nil;
                  if (mcpcour = nil)
                  then begin
                              mcpcour := newmcp;
                              fmcp := newmcp
                        end
                  else begin
                              mcpcour^.suivant := newmcp;
                              mcpcour := mcpcour^.suivant
                        end
            end;
            if (lignefconttxt[length(lignefconttxt)] = '.') then finphrase := true
      end;
      close (fconttxt)
end;

(*************************************************************************)

procedure FIND_INDEX;

(* sp{cifications : FIND_INDEX re\oit le mot 'mot' et trouve la position     *)
(*                  'where' de la 1[ analyse morphologique possible de ce mot*)
(*                  dans le fichier des analyses morphologiques.             *)

var
    min, max, cour : integer;
    ok : boolean;

begin
      assign (findexfamml, 'b:index.am');
      reset (findexfamml);
      min := 0;
      max := ( filesize (findexfamml) - 1 );
      ok := false;
```

```
      while ( (ok = false) and (min <= max) ) do
      begin
            cour := (min + max) div 2;
            seek (findexfamml, cour);
            read (findexfamml, ligneindex);
            if ( ligneindex.motlatin = mot )
            then begin
                      where := ligneindex.pos;
                      ok := true
                 end
            else if ( ligneindex.motlatin < mot )
                 then min := cour + 1
                 else max := cour - 1
      end;
      close (findexfamml)
end;

(*****************************************************************************)

procedure COPY_FA_LA;

var
    newlna, lnacour, oldlna, fauxnoeudfrere : listnoeudaa;
    oldfna : fnoeudaa;

begin
      new (fauxnoeudfrere);
      fauxnoeudfrere^.noeudfils := nil;
      fauxnoeudfrere^.noeudfrere := nil;
      fauxnoeudfrere^.noeudpere := nil;
      with fauxnoeudfrere^.noeud do
      begin
           ind_brk := break;
           appart_analyse_correct := false;
           comment_syst := '';
           comment_specif := '';
           accept_comment := false;
           typenoeud := n1;
           fonction := sujet;
           mot1 := 'faux noeud frere !'
      end;
      assign (fna, nomfichanal);
      reset (fna);
      lnacour := nil;
      while ( not eof(fna) ) do
      begin
            read (fna, fnaligne);
            new (newlna);
            newlna^.noeud := fnaligne.noeud;
            if (fnaligne.noeudfils = false) then newlna^.noeudfils := nil;
            if (fnaligne.noeudfrere = false) then newlna^.noeudfrere := nil
                                             else newlna^.noeudfrere := fauxnoeudfrere;
            if (lnacour = nil)
            then begin
                      lnacour := newlna;
                      newlna^.noeudpere := nil;
                      firstlna := newlna
                 end
            else if (oldfna.noeudfils = true)
                 then begin
                           oldlna^.noeudfils := newlna;
                           newlna^.noeudpere := oldlna;
                           lnacour := newlna
                      end
                 else begin
```

```
                        while (oldlna^.noeudfrere <> fauxnoeudfrere)
                        do oldlna := oldlna^.noeudpere;
                        oldlna^.noeudfrere := newlna;
                        newlna^.noeudpere := oldlna^.noeudpere;
                        lnacour := newlna
                    end;
            oldfna := fnaligne;
            oldlna := newlna
      end;
      close (fna)
end;

(*************************************************************************)

procedure COPY_LA_FA;

var
    fin : boolean;
    lnacour : listnoeudaa;

begin
      lnacour := firstlna;
      assign (fna, nomfichanal);
      rewrite (fna);
      fin := false;
      while ( (lnacour <> nil) and (fin = false) ) do
      begin
            if (lnacour^.noeudfils <> nil) then fnaligne.noeudfils := true
                                           else fnaligne.noeudfils := false;
            if (lnacour^.noeudfrere <> nil) then fnaligne.noeudfrere := true
                                            else fnaligne.noeudfrere := false;
            fnaligne.noeud := lnacour^.noeud;
            write (fna, fnaligne);
            while ( lnacour^.noeudfils <> nil ) do
            begin
                  lnacour := lnacour^.noeudfils;
                  if (lnacour^.noeudfils <> nil) then fnaligne.noeudfils := true
                                                 else fnaligne.noeudfils := false;
                  if (lnacour^.noeudfrere <> nil) then fnaligne.noeudfrere := true
                                                  else fnaligne.noeudfrere := false;
                  fnaligne.noeud := lnacour^.noeud;
                  write (fna, fnaligne)
            end;
            if (lnacour^.noeudfrere <> nil)
            then lnacour := lnacour^.noeudfrere
            else begin
                        lnacour := lnacour^.noeudpere;
                        while ( (lnacour <> nil) and (lnacour^.noeudfrere = nil) )
                        do lnacour := lnacour^.noeudpere;
                        if (lnacour <> nil) then lnacour := lnacour^.noeudfrere
                                            else fin := true
                 end
      end;
      close (fna)
end;

(*************************************************************************)

function VALID_TXT ;

(* specifications : si 'b:nametxt.fct' correspond au titre d'un texte latin existant *)
(*                  alors VALID_TXT prend la valeur vrai                             *)
(*                  sinon VALID_TXT prend la valeur faux                             *)
```

```
begin
     nametxt := concat ('b:', nametxt, '.fct' );
(*$I-*)
     assign (fconttxt, nametxt);
     reset (fconttxt);
     if (ioresult = 0) then begin
                                valid_txt := true;
                                close (fconttxt)
                           end
                      else valid_txt := false
(*$I+*)
end;

(*************************************************************************)

procedure LIST_TXT_LAT;

(* sp(cifications : LIST_TXT_LAT fournit la liste des fichiers '*.fct' se   *)
(*                  trouvant sur le drive 'b';                              *)
(*                  ces fichiers contiennent chacun un texte latin.         *)


var dirinfo         : searchrec;
    ymax, k, iocode : integer;
    ch              : char;
    titre           : str30;
    namefdt         : str14;

begin
     clrscr;
     gotoxy (7,2);
     writeln ('LISTE DES FICHIERS CONTENANT DES TEXTES LATINS SUR LE DRIVE ''B'' :');
     writeln;
     writeln;
     ymax := 23;
     findfirst ('b:*.fct', anyfile, dirinfo);
     while (doserror = 0) do
     begin
          if (wherey > ymax)
          then begin
                    gotoxy (35,25);
                    write ('Tapez une touche pour voir la page suivante');
                    repeat until keypressed;
                    ch := readkey;
                    clrscr;
                    gotoxy (7, 2);
                    writeln ('LISTE DES FICHIERS CONTENANT DES TEXTES LATINS SUR LE DRIVE ''B'' :');
                    writeln;
                    writeln
               end;
          write (dirinfo.name);
          k := length(dirinfo.name) - 4;
          namefdt := copy (dirinfo.name, 1, k);
          namefdt := concat ('b:', namefdt, '.dsc');
(*$I-*)
          assign (fdescrtxt, namefdt);
          reset (fdescrtxt);
          iocode := ioresult;
          if (iocode <> 0)
          then titre := ''
          else begin
                    read (fdescrtxt, lignefdt);
                    titre := lignefdt.titre_txt;
                    close (fdescrtxt)
               end;
```

```
(*$I+*)
          gotoxy (25, wherey);
          writeln (titre);
          findnext (dirinfo)
     end;
     gotoxy (35,25);
     write ('Tapez une touche pour retourner au menu');
     repeat until keypressed;
     ch := readkey
end;

(*************************************************************************)


procedure VISU_TXT_LAT ;

(* sp{cifications : VISU_TXT_LAT permet de visualiser @ l'{cran un texte latin existant. *)
(*                  Si le texte est trop long que pour apparai^tre en un {cran,          *)
(*                  la pression d'une touche au clavier permettra d'en voir la suite .   *)
(*                  De me^me, @ la fin du texte, il suffira d'appuyer sur une touche au  *)
(*                  clavier pour retourner au menu.                                      *)

var
    ch         : char;
    iocode, k  : integer;
    namefdt    : str14;
    titre      : str60;
    resultat   : boolean;

begin
     k := length(namefcttxt) - 4;
     namefdt := copy (namefcttxt, 1, k);
     namefdt := concat (namefdt, '.dsc');
(*$I-*)
     assign (fdescrtxt, namefdt);
     reset (fdescrtxt);
     iocode := ioresult;
     if (iocode <> 0)
     then titre := ''
     else begin
               read (fdescrtxt, lignefdt);
               titre := lignefdt.titre_txt;
               close (fdescrtxt)
          end;
(*$I+*)
     VISUTXTPHR (namefcttxt, 0, titre, 2, 23, resultat);
     gotoxy (35,25);
     write ('Tapez une touche pour retourner au menu');
     repeat until keypressed;
     ch := readkey;
end;

(*************************************************************************)

procedure LIST_RES_TRT_AUT ;

(* sp{cifications : LIST_RES_TRT_AUT fournit le r{sultat du traitement      *)
(*                  automatique de chaque phrase du texte latin dont la     *)
(*                  description est contenue dans le fichier 'namefdt'.     *)

var
    titre : str30;
    ymax  : integer;
    ch    : char;
```

```
begin
     clrscr;
     assign (fdescrtxt, namefdt);
     reset (fdescrtxt);
     read (fdescrtxt, lignefdt);
     titre := lignefdt.titre_txt;
     gotoxy (24, 2);
     writeln ('RESULTATS DU TRAITEMENT AUTOMATIQUE');
     gotoxy (26, 4);
     writeln (titre);
     writeln;
     writeln;
     ymax := 23;
     write ('r{sultat global du traitement automatique du texte : ');
     case lignefdt.res_trt_aut of
          nontrtaut   : writeln ('non encore trait{');
          incoherence : writeln ('incoh{rence(s)');
          ok          : writeln ('correct')
     end;
     writeln;
     while (not eof (fdescrtxt) ) do
     begin
          read (fdescrtxt, lignefdt);
          if (wherey > ymax)
          then begin
                    gotoxy (35,25);
                    write ('Tapez une touche pour voir la page suivante');
                    repeat until keypressed;
                    ch := readkey;
                    clrscr;
                    gotoxy (24, 2);
                    writeln ('RESULTATS DU TRAITEMENT AUTOMATIQUE');
                    gotoxy (26, 4);
                    writeln (titre);
                    writeln;
                    writeln
               end;
          write ('r{sultat du traitement automatique de la phrase ', lignefdt.numero_phrase, '  : ');
          case lignefdt.res_trt_aut_phrase of
               incoherente : writeln ('incoh{rence(s)');
               coherente   : writeln ('correcte');
               nontrt      : writeln ('non encore trait{e')
          end
     end;
     gotoxy (35,25);
     write ('Tapez une touche pour retourner au menu');
     repeat until keypressed;
     ch := readkey;
     close (fdescrtxt)
end;

(*************************************************************************)

procedure VISU_ETAT_PREP_TXT_LAT ;

(* sp{cifications : VISU_ETAT_PREP_TXT_LAT fournit l'{tat de pr{paration de  *)
(*                  chaque phrase du texte latin dont la description est     *)
(*                  contenue dans le fichier 'namefdt'.                      *)

var
    titre : str30;
    ymax  : integer;
    ch    : char;

begin
```

```
    clrscr;
    assign (fdescrtxt, namefdt);
    reset (fdescrtxt);
    read (fdescrtxt, lignefdt);
    titre := lignefdt.titre_txt;
    gotoxy (24, 2);
    writeln ('ETAT DE PREPARATION DE L''ANALYSE');
    gotoxy (26, 4);
    writeln (titre);
    writeln;
    writeln;
    ymax := 23;
    write ('{tat de pr{paration global du texte : ');
    case lignefdt.etat_prep_txt of
        pret   : writeln ('pre^t');
        enprep : writeln ('en pr{paration')
    end;
    writeln;
    while (not eof (fdescrtxt) ) do
    begin
        read (fdescrtxt, lignefdt);
        if (wherey > ymax)
        then begin
                gotoxy (35,25);
                write ('Tapez une touche pour voir la page suivante');
                repeat until keypressed;
                ch := readkey;
                clrscr;
                gotoxy (24, 2);
                writeln ('ETAT DE PREPARATION DE L''ANALYSE');
                gotoxy (26, 4);
                writeln (titre);
                writeln;
                writeln
            end;
        write ('{tat de pr{paration de phrase ', lignefdt.numero_phrase, '     : ');
        case lignefdt.etat_prep_phrase of
            e1 : writeln ('analyse fournie par le LASLA de Li}ge');
            e2 : writeln ('compl{ments manuels en cours');
            e3 : writeln ('compl{ments manuels finis, non v{rifi{s');
            e4 : writeln ('compl{ments manuels incoh{rents');
            e5 : writeln ('analyse coh{rente, pre^te')
        end
    end;
    gotoxy (35,25);
    write ('Tapez une touche pour retourner au menu');
    repeat until keypressed;
    ch := readkey;
    close (fdescrtxt)
end;


End.
```

```
Unit CPLTPROC;

Interface
uses (*$U b:globald2 *) GLOBALDECL2, CRT, DOS;

procedure GESTCOMM (ligne : integer; var ldcour : list_descr; texteb : boolean);

procedure AFFCOMM (ligne : integer; var comment : str80; existfna : boolean);

procedure AFFACCCOMM (ligne : integer; var accept : boolean ; existfna : boolean);

procedure AFFINDBRK (ligne : integer; var lnoeudaa : listnoeudaa; existfna : boolean ; var chemin : boolean);

procedure REMPLI_AM_V (var noeud : noeudaa; var ligne : analmorphmotlat);

procedure AFFICH_AM_V (ligne : integer; var noeud : noeudaa);

procedure REMPLI_AM_SUBST (var noeud : noeudaa; var ligne : analmorphmotlat);

procedure AFFICH_AM_SUBST (ligne : integer; var noeud : noeudaa);

procedure REMPLI_AM_ADJ (var noeud : noeudaa; var ligne : analmorphmotlat);

procedure AFFICH_AM_ADJ (ligne : integer; var noeud : noeudaa);

procedure REMPLI_AM_ADJPRON (var noeud : noeudaa; var ligne : analmorphmotlat);

procedure AFFICH_AM_ADJPRON (ligne : integer; var noeud : noeudaa);

procedure REMPLI_AM_CONJ (var noeud : noeudaa; var ligne : analmorphmotlat);

procedure AFFICH_AM_CONJ (ligne : integer; var noeud : noeudaa);

procedure REMPLI_AM_PREP (var noeud : noeudaa; var ligne : analmorphmotlat);

procedure AFFICH_AM_PREP (ligne : integer; var noeud : noeudaa);

procedure AFFICHN3 (var noeud : noeudaa; var firstmcp : lcontphr; miny : integer; maxy : integer);

procedure AFFICHLN3 (var firstln3 : lcontphr; miny : integer; maxy : integer);

procedure REMPLI_N3_LN3 (var firstln3 : lcontphr; var ensmots3 : arrbool85; var firstmcp : lcontphr);

procedure TRSF_N3_LN3 (var firstln3 : lcontphr; var ensmots3 : arrbool85; var firstmcp : lcontphr);

procedure MAJLIBRE (var firstmcp : lcontphr; var noeud : noeudaa);

procedure MAJLNACOUR (var lnacour : listnoeudaa; var existfna : boolean; var firstmcp : lcontphr;
                      var remonte : boolean; var fin : boolean);


Implementation

procedure GESTCOMM;

(* sp(cifications : GESTCOMM affiche @ partir de la ligne 'ligne',          *)
(*                  'ldcour^.contenu.comment_txt' ou 'ldcour^.contenu.comment_phrase *)
(*                  selon la valeur de 'texteb' et propose de le modifier,  *)
(*                  garder ou annuler.                                      *)
var
   ch : char;
   ok : boolean;
   k : integer;
```

```
begin
     gotoxy (1, ligne);
     if (texteb = true) then begin
                                writeln ('Commentaire de ce texte :');
                                write (ldcour^.contenu.comment_txt)
                            end
                       else begin
                                writeln ('Commentaire de cette phrase :');
                                write (ldcour^.contenu.comment_phrase)
                            end;
     gotoxy (1,ligne + 3);
     write ('Voulez-vous ''A''nnuler, ''M''odifier ou ''G''arder ce commentaire ? _');
     gotoxy (wherex - 1, wherey);
     ok := false;
     while (ok = false) do
     begin
          if keypressed
          then begin
                    ch := readkey;
                    case ch of
                         'A', 'a', 'M', 'm', 'G', 'g' : begin
                                                             ok := true;
                                                             write (ch)
                                                        end;
                         else begin
                                   sound (220);
                                   delay (200);
                                   nosound
                              end
                    end
               end
     end;
     case ch of
          'A', 'a' : begin
                          if (texteb = true) then ldcour^.contenu.comment_txt := ''
                                             else ldcour^.contenu.comment_phrase := '';
                          gotoxy (1, ligne + 1);
                          for k := 1 to 80 do write (' ');
                          gotoxy (1, ligne + 1);
                          if (texteb = true) then write (ldcour^.contenu.comment_txt)
                                             else write (ldcour^.contenu.comment_phrase);
                          gotoxy (45, ligne + 4);
                          write ('Tapez une touche pour continuer !');
                          repeat until keypressed;
                          ch := readkey
                     end;
          'G', 'g' : begin
                          gotoxy (45, 25);
                          write ('Tapez une touche pour continuer !');
                          repeat until keypressed;
                          ch := readkey
                     end;
          'M', 'm' : begin
                          if (texteb = true) then ldcour^.contenu.comment_txt := ''
                                             else ldcour^.contenu.comment_phrase := '';
                          gotoxy (1, ligne + 1);
                          for k := 1 to 80 do write ('_');
                          gotoxy (1, ligne + 1);
                          if (texteb = true) then readln (ldcour^.contenu.comment_txt)
                                             else readln (ldcour^.contenu.comment_phrase);
                          gotoxy (45, ligne + 4);
                          write ('Tapez une touche pour continuer !');
                          repeat until keypressed;
                          ch := readkey
                     end
```

```
    end
end;

(*-------------------------------------------------------------------------*)

procedure AFFCOMM;

(* sp{cifications : si existfna = true, AFFCOMM affiche @ la ligne 'ligne+1' *)
(*                    comment et offre la possibilit{ de le modifier,       *)
(*                  si existfna = false, AFFCOMM demande l'introduction de  *)
(*                    comment @ la ligne 'ligne+1'.                         *)

var
   ok : boolean;
   ch : char;
   k  : integer;

begin
     gotoxy (1, ligne);
     writeln ('Votre commentaire :');
     if (existfna = true)
     then begin
               writeln (comment);
               write ('Voulez-vous modifier votre commentaire ? ''O''ui, ''N''on : _');
               gotoxy (wherex - 1, wherey);
               ok := false;
               while (ok = false) do
               begin
                    if keypressed
                     then begin
                               ch := readkey;
                               case ch of
                                    'O', 'o' : begin
                                                    write (ch);
                                                    gotoxy (1, ligne + 1);
                                                    for k := 1 to 80 do write ('_');
                                                    gotoxy (1, ligne + 1);
                                                    readln (comment);
                                                    ok := true
                                               end;
                                    'N', 'n' : begin
                                                    write (ch);
                                                    ok := true
                                               end;
                                    else       begin
                                                    sound (220);
                                                    delay (200);
                                                    nosound
                                               end
                               end
                          end
               end
          end
      else begin
               for k := 1 to 80 do write ('_');
               gotoxy (1, ligne + 1);
               readln (comment)
          end
end;

(*-------------------------------------------------------------------------*)

procedure AFFACCCOMM;

(* sp{cifications : si existfna = true, AFFACCCOMM affiche @ la ligne 'ligne+1' *)
```

```
(*                  la mention d'acceptation selon accept et offre la      *)
(*                  possibilit( de modifier cette acceptation,             *)
(*               si existfna = false, AFFCOMM demande l'introduction de    *)
(*                  l'acceptation accept @ la ligne 'ligne+1'.             *)

var
   ok : boolean;
   ch : char;

begin
     gotoxy (1, ligne);
     write ('Acceptez-vous l''affichage du commentaire : ''O''ui, ''N''on : _');
     gotoxy (wherex - 1, wherey);
     if (existfna = true)
     then begin
               case accept of
                    true : write ('O ? ');
                    false : write ('N ? ')
               end;
               ok := false;
               while (ok = false) do
               begin
                    if keypressed
                    then begin
                              ch := readkey;
                              case ch of
                                   'O', 'o' : begin
                                                   write (ch);
                                                   accept := true;
                                                   ok := true
                                              end;
                                   'N', 'n' : begin
                                                   write (ch);
                                                   accept := false;
                                                   ok := true
                                              end;
                  (* return *)       #13      : ok := true;
                                   else       begin
                                                   sound (220);
                                                   delay (200);
                                                   nosound
                                              end
                              end
                         end
               end
          end
     else begin
               ok := false;
               while (ok = false) do
               begin
                    if keypressed
                    then begin
                              ch := readkey;
                              case ch of
                                   'O', 'o' : begin
                                                   write (ch);
                                                   accept := true;
                                                   ok := true
                                              end;
                                   'N', 'n' : begin
                                                   write (ch);
                                                   accept := false;
                                                   ok := true
                                              end;
                                   else       begin
```

```
                                        sound (220);
                                        delay (200);
                                        nosound
                                    end
                                end
                            end
                        end
                end;

(*-------------------------------------------------------------------------*)

procedure AFFINDBRK;

(* sp(cifications : si existfna = true, AFFINDBRK affiche @ la ligne 'ligne+1' *)
(*                  la mention du type d'{tape d'analyse selon lnoeudaa^.noeud.ind_brk  *)
(*                  et offre la possibilit{ de modifier ce renseignement,   *)
(*                si existfna = false, AFFINDBRK demande l'introduction de   *)
(*                  cette mention lnoeudaa^.noeud.ind_brk @ la ligne ligne+1*)
(*                chemin est une information {ventuellement modifi{e,        *)
(*                permettant de mettre @ jour la mention d'appartenance @ une*)
(*                analyse correcte des ance^tres du noeud.                   *)

var
    ok, modif : boolean;
    ch : char;
    lnapere : listnoeudaa;

begin
     modif := false;
     gotoxy (1, ligne);
     write ('Noeud ''I''ndicatif ou ''B''reak : _');
     gotoxy (wherex - 1, wherey);
     if (existfna = true)
     then begin
               case lnoeudaa^.noeud.ind_brk of
                    indicatif : write ('I ? ');
                    break : write ('B ? ')
               end;
               ok := false;
               while (ok = false) do
               begin
                    if keypressed
                    then begin
                              ch := readkey;
                              case ch of
                                   'I', 'i' : begin
                                                   write (ch);
                                                   if (lnoeudaa^.noeud.ind_brk = break) then modif := true;
                                                   lnoeudaa^.noeud.ind_brk := indicatif;
                                                   ok := true
                                              end;
                                   'B', 'b' : begin
                                                   write (ch);
                                                   if (lnoeudaa^.noeud.ind_brk = indicatif)
                                                   then begin
                                                             modif := true;
                                                             lnoeudaa^.noeudfils := nil
                                                        end;
                                                   lnoeudaa^.noeud.ind_brk := break;
                                                   ok := true
                                              end;
                              (* return *)     #13      : ok := true;
                                   else       begin
                                                   sound (220);
```

```
                                                delay (200);
                                                nosound
                                            end
                                    end
                                end
                            end
                        else begin
                                modif := true;
                                ok := false;
                                while (ok = false) do
                                begin
                                    if keypressed
                                    then begin
                                            ch := readkey;
                                            case ch of
                                                'I', 'i' : begin
                                                            write (ch);
                                                            lnoeudaa^.noeud.ind_brk := indicatif;
                                                            ok := true
                                                        end;
                                                'B', 'b' : begin
                                                            write (ch);
                                                            lnoeudaa^.noeud.ind_brk := break;
                                                            ok := true
                                                        end;
                                                else      begin
                                                            sound (220);
                                                            delay (200);
                                                            nosound
                                                        end
                                            end
                                        end
                                end
                            end;
                        if (modif = true)
                        then begin
                                if (lnoeudaa^.noeud.ind_brk = indicatif)
                                then begin
                                        chemin := true;
                                        lnoeudaa^.noeud.appart_analyse_correct := true;
                                        lnapere := lnoeudaa^.noeudpere;
                                        while (lnapere <> nil) do
                                        begin
                                            lnapere^.noeud.appart_analyse_correct := true;
                                            lnapere := lnapere^.noeudpere
                                        end
                                    end
                                else begin
                                        lnoeudaa^.noeud.appart_analyse_correct := false;
                                        if (chemin = false)
                                        then begin
                                                lnapere := lnoeudaa^.noeudpere;
                                                while (lnapere <> nil) do
                                                begin
                                                    lnapere^.noeud.appart_analyse_correct := false;
                                                    lnapere := lnapere^.noeudpere
                                                end
                                            end
                                    end
                            end
                    end
end;

(*-------------------------------------------------------------------------*)
```

```
procedure REMPLI_AM_V;

(* sp(cifications : @ partir des donn(es enregistr(es dans ligne, REMPLI_AM_V*)
(*                  remplit 'noeud'(= analyse morphologique verbale).       *)

begin
     case ligne.sscatgram_dgr_vx of
          '1', 'A', 'J'      : noeud.conjv := c1;
          '2', 'B', 'K', 'S' : noeud.conjv := c2;
          '3', 'C', 'L', 'T' : noeud.conjv := c3;
          '4', 'D', 'M'      : noeud.conjv := c4;
          '5', 'E', 'N'      : noeud.conjv := c5;
          '6', 'F', 'O'      : noeud.conjv := c6
     end;
     case ligne.sscatgram_dgr_vx of
          '1', '2', '3', '4', '5', '6' : noeud.voixv := actif;
          'A', 'B', 'C', 'D', 'E', 'F' : noeud.voixv := passif;
          'J', 'K', 'L', 'M', 'N', 'O' : noeud.voixv := deponent;
          'S', 'T'                     : noeud.voixv := semideponent
     end;
     case ligne.mode of
          '1', '2', '3', '7', '8', '9' :
               case lignefamml.cas_pers_nbr of
                    'A' : begin
                               noeud.persv := p1;
                               noeud.nbr1v := sing
                          end;
                    'B' : begin
                               noeud.persv := p2;
                               noeud.nbr1v := sing
                          end;
                    'C' : begin
                               noeud.persv := p3;
                               noeud.nbr1v := sing
                          end;
                    'J' : begin
                               noeud.persv := p1;
                               noeud.nbr1v := pluriel
                          end;
                    'K' : begin
                               noeud.persv := p2;
                               noeud.nbr1v := pluriel
                          end;
                    'L' : begin
                               noeud.persv := p3;
                               noeud.nbr1v := pluriel
                          end;
                     else begin noeud.persv := pnul;
                               noeud.nbr1v := nombrenul
                          end
               end;
          '4', '5', '6' :
               begin
               case ligne.cas_pers_nbr of
                    'A', 'J' : noeud.casv := nom;
                    'B', 'K' : noeud.casv := voc;
                    'C', 'L' : noeud.casv := acc;
                    'D', 'M' : noeud.casv := gen;
                    'E', 'N' : noeud.casv := datif;
                    'F', 'O' : noeud.casv := abl;
                    'G', 'P' : noeud.casv := loc;
                    'Z'      : begin
                                    noeud.casv := indecl;
                                    noeud.nbr2v := nombrenul
                               end
```

```
                end;
            case ligne.cas_pers_nbr of
                'A', 'B', 'C', 'D', 'E', 'F', 'G' : noeud.nbr2v := sing;
                'J', 'K', 'L', 'M', 'N', 'O', 'P' : noeud.nbr2v := pluriel
                end
            end
    end;
    case ligne.mode of
        '1' : noeud.modev := indic;
        '2' : noeud.modev := imp;
        '3' : noeud.modev := subj;
        '4' : noeud.modev := part;
        '5' : noeud.modev := adjverb;
        '6' : noeud.modev := gerondif;
        '7' : noeud.modev := inf;
        '8' : noeud.modev := supinum;
        '9' : noeud.modev := supinu
    end;
    case ligne.temps of
        '1' : noeud.tempsv := pres;
        '2' : noeud.tempsv := impft;
        '3' : noeud.tempsv := futspl;
        '4' : noeud.tempsv := prft;
        '5' : noeud.tempsv := plusqueprft;
        '6' : noeud.tempsv := futant;
        '7', '8', '9' : noeud.tempsv := fuisse;
        else noeud.tempsv := tempsnul
    end;
    case ligne.fonction of
        '1' : noeud.fonctionv := princ;
        '2' : noeud.fonctionv := subord;
        else noeud.fonctionv := fonctionnul
    end;
    case ligne.genre of
        '1' : noeud.genrev := commun;
        '2' : noeud.genrev := fem;
        '3' : noeud.genrev := mascfem;
        '4' : noeud.genrev := masc;
        '5' : noeud.genrev := mascneutre;
        '6' : noeud.genrev := neutre;
        else  noeud.genrev := genrenul
    end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_V;

(* sp(cifications : AFFICH_AM_V affiche @ partir de la ligne 'ligne' jusqu'@ *)
(*                  la ligne 'ligne+3', les informations contenues dans     *)
(*                  'noeud' concernant une analyse morphologique verbale .  *)

begin
    gotoxy (1, ligne);
    write ('Conjugaison de ce verbe : ');
    case noeud.conjv of
        c1 : write ('1[');
        c2 : write ('2[');
        c3 : write ('3[');
        c4 : write ('4[');
        c5 : write ('4[ bis');
        c6 : write ('anomal')
    end;
    gotoxy (40, ligne);
    write ('Voix : ');
```

```
case noeud.voixv of
     actif        : write ('actif');
     passif       : write ('passif');
     deponent     : write ('deponent');
     semideponent : write ('semideponent')
end;
case noeud.modev of
     indic, imp, subj, inf, supinum, supinu :
                  begin
                       gotoxy (1, ligne+1);
                       write ('Personne : ');
                       case noeud.persv of
                            p1   : write ('1[');
                            p2   : write ('2[');
                            p3   : write ('3[');
                            pnul : write ('-')
                       end;
                       gotoxy (40, ligne+1);
                       write ('Nombre : ');
                       case noeud.nbr1v of
                            sing      : write ('singulier');
                            pluriel   : write ('pluriel');
                            nombrenul : write ('-')
                       end
                  end;
     part, adjverb, gerondif :
                  begin
                       gotoxy (1, ligne+1);
                       write ('Cas : ');
                       case noeud.casv of
                            nom    : write ('nominatif');
                            voc    : write ('vocatif');
                            acc    : write ('accusatif');
                            gen    : write ('g{nitif');
                            datif  : write ('datif');
                            abl    : write ('ablatif');
                            loc    : write ('locatif');
                            indecl : write ('ind{clinable')
                       end;
                       gotoxy (40, ligne+1);
                       write ('Nombre : ');
                       case noeud.nbr2v of
                            sing      : write ('singulier');
                            pluriel   : write ('pluriel');
                            nombrenul : write ('-')
                       end
                  end
end;
gotoxy (1, ligne+2);
write ('Mode : ');
case noeud.modev of
     indic     : write ('indicatif');
     imp       : write ('imp{ratif');
     subj      : write ('subjonctif');
     part      : write ('participe');
     adjverb   : write ('adjectif verbal');
     gerondif  : write ('g{rondif');
     inf       : write ('infinitif');
     supinum   : write ('supin en ''um''');
     supinu    : write ('supin en ''u''')
end;
gotoxy (40, ligne+2);
write ('Temps : ');
case noeud.tempsv of
     pres        : write ('pr{sent');
```

```
          impft       : write ('imparfait');
          futspl      : write ('futur simple');
          prft        : write ('parfait');
          plusqueprft : write ('plus que parfait');
          futant      : write ('futur ant{rieur');
          fuisse      : write ('''fuisse''');
          tempsnul    : write ('-')
     end;
     gotoxy (1, ligne+3);
     write ('Genre : ');
     case noeud.genrev of
          commun     : write ('commun');
          fem        : write ('f{minin');
          mascfem    : write ('masculin-f{minin');
          masc       : write ('masculin');
          mascneutre : write ('masculin-neutre');
          neutre     : write ('neutre');
          genrenul   : write ('-')
     end;
     gotoxy (40, ligne+3);
     write ('Fonction : ');                    (* ? *)
     case noeud.fonctionv of
          subord     : write ('subordonn{e');
          princ      : write ('principal');
          fonctionnul :  write ('-')
     end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_SUBST;

(* sp{cifications : @ partir des donn{es enregistr{es dans ligne, REMPLI_AM_SUBST*)
(*                  remplit 'noeud'(= analyse morphologique d'un substantif).   *)

begin
     case ligne.sscatgram_dgr_vx of
          '1' : noeud.declsubst := d1;
          '2' : noeud.declsubst := d2;
          '3' : noeud.declsubst := d3;
          '4' : noeud.declsubst := d4;
          '5' : noeud.declsubst := d5;
          '6' : noeud.declsubst := d6;   (* anomal *)
          '7' : noeud.declsubst := d7;   (* d{cl.gr. *)
     end;
     case ligne.cas_pers_nbr of
          'A', 'J' : noeud.cassubst := nom;
          'B', 'K' : noeud.cassubst := voc;
          'C', 'L' : noeud.cassubst := acc;
          'D', 'M' : noeud.cassubst := gen;
          'E', 'N' : noeud.cassubst := datif;
          'F', 'O' : noeud.cassubst := abl;
          'G', 'P' : noeud.cassubst := loc;
          'Z'      : begin
                          noeud.cassubst := indecl;
                          noeud.nbrsubst := nombrenul
                     end
     end;
     case ligne.cas_pers_nbr of
          'A', 'B', 'C', 'D', 'E', 'F', 'G' : noeud.nbrsubst := sing;
          'J', 'K', 'L', 'M', 'N', 'O', 'P' : noeud.nbrsubst := pluriel
     end;
     case ligne.genre of
          '1' : noeud.genresubst := commun;
          '2' : noeud.genresubst := fem;
```

```
          '3' : noeud.genresubst := mascfem;
          '4' : noeud.genresubst := masc;
          '5' : noeud.genresubst := mascneutre;
          '6' : noeud.genresubst := neutre;
          else  noeud.genresubst := genrenul
     end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_SUBST;

(* sp(cifications : AFFICH_AM_SUBST affiche @ partir de la ligne 'ligne' jusqu'@ *)
(*                  la ligne 'ligne+3', les informations contenues dans        *)
(*                  'noeud' concernant une analyse morphologique d'un substantif.*)

begin
     gotoxy (1, ligne);
     write ('D(clinaison de ce substantif : ');
     case noeud.declsubst of
          d1 : write ('1[');
          d2 : write ('2[');
          d3 : write ('3[');
          d4 : write ('4[');
          d5 : write ('5[');
          d6 : write ('anomal');
          d7 : write ('d(cl.gr.')
     end;
     gotoxy (1, ligne+1);
     write ('Cas : ');
     case noeud.cassubst of
          nom    : write ('nominatif');
          voc    : write ('vocatif');
          acc    : write ('accusatif');
          gen    : write ('g(nitif');
          datif  : write ('datif');
          abl    : write ('ablatif');
          loc    : write ('locatif');
          indecl : write ('ind(clinable')
     end;
     gotoxy (1, ligne+2);
     write ('Nombre : ');
     case noeud.nbrsubst of
          sing      : write ('singulier');
          pluriel   : write ('pluriel');
          nombrenul : write ('-')
     end;
     gotoxy (1, ligne+3);
     write ('Genre : ');
     case noeud.genresubst of
          commun     : write ('commun');
          fem        : write ('f(minin');
          mascfem    : write ('masculin-f(minin');
          masc       : write ('masculin');
          mascneutre : write ('masculin-neutre');
          neutre     : write ('neutre');
          genrenul   : write ('-')
     end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_ADJ;

(* sp(cifications : @ partir des donn(es enregistr(es dans ligne, REMPLI_AM_ADJ*)
```

```
(*                remplit 'noeud'(= analyse morphologique d'un adjectif).  *)

begin
     case ligne.sscatgram_dgr_vx of
          '1', 'A', 'J' : noeud.classeadj := a1;
          '2', 'B', 'K' : noeud.classeadj := a2;
          '3', 'C', 'L' : noeud.classeadj := a3;
          '4', 'D', 'M' : noeud.classeadj := a4;
          '5', 'E', 'N' : noeud.classeadj := a5;
          '6', 'F', 'O' : noeud.classeadj := a6;
          '7'           : noeud.classeadj := a7;
     end;
     case ligne.sscatgram_dgr_vx of
          '1', '2', '3', '4', '5', '6', '7' : noeud.degreadj := positif;
          'A', 'B', 'C', 'D', 'E', 'F'      : noeud.degreadj := comparatif;
          'J', 'K', 'L', 'M', 'N', 'O'      : noeud.degreadj := superlatif
     end;
     case ligne.cas_pers_nbr of
          'A', 'J' : noeud.casadj := nom;
          'B', 'K' : noeud.casadj := voc;
          'C', 'L' : noeud.casadj := acc;
          'D', 'M' : noeud.casadj := gen;
          'E', 'N' : noeud.casadj := datif;
          'F', 'O' : noeud.casadj := abl;
          'G', 'P' : noeud.casadj := loc;
          'Z'      : begin
                          noeud.casadj := indecl;
                          noeud.nbradj := nombrenul
                     end
     end;
     case ligne.cas_pers_nbr of
          'A', 'B', 'C', 'D', 'E', 'F', 'G' : noeud.nbradj := sing;
          'J', 'K', 'L', 'M', 'N', 'O', 'P' : noeud.nbradj := pluriel
     end;
     case ligne.genre of
          '1' : noeud.genreadj := commun;
          '2' : noeud.genreadj := fem;
          '3' : noeud.genreadj := mascfem;
          '4' : noeud.genreadj := masc;
          '5' : noeud.genreadj := mascneutre;
          '6' : noeud.genreadj := neutre;
          else  noeud.genreadj := genrenul
     end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_ADJ;

(* sp{cifications : AFFICH_AM_ADJ affiche @ partir de la ligne 'ligne' jusqu'@ *)
(*                  la ligne 'ligne+3', les informations contenues dans       *)
(*                  'noeud' concernant une analyse morphologique d'un adjectif.*)

begin
     gotoxy (1, ligne);
     write ('Classe de cet adjectif : ');
     case noeud.classeadj of
          a1 : write ('1[');
          a2 : write ('2[ cons.');
          a3 : write ('2[ -er');
          a4 : write ('2[ -is');
          a5 : write ('2[ imp.');
          a6 : write ('anomal');
          a7 : write ('d{cl.gr.')
     end;
```

```
      gotoxy (1, ligne + 1);
      write ('Degr{ : ');
      case noeud.degreadj of
           positif : write ('positif');
           comparatif : write ('comparatif');
           superlatif : write ('superlatif')
      end;
      gotoxy (1, ligne+2);
      write ('Cas : ');
      case noeud.casadj of
           nom    : write ('nominatif');
           voc    : write ('vocatif');
           acc    : write ('accusatif');
           gen    : write ('g{nitif');
           datif  : write ('datif');
           abl    : write ('ablatif');
           loc    : write ('locatif');
           indecl : write ('ind{clinable')
      end;
      gotoxy (40, ligne+2);
      write ('Nombre : ');
      case noeud.nbradj of
           sing      : write ('singulier');
           pluriel   : write ('pluriel');
           nombrenul : write ('-')
      end;
      gotoxy (1, ligne+3);
      write ('Genre : ');
      case noeud.genreadj of
           commun     : write ('commun');
           fem        : write ('f{minin');
           mascfem    : write ('masculin-f{minin');
           masc       : write ('masculin');
           mascneutre : write ('masculin-neutre');
           neutre     : write ('neutre');
           genrenul   : write ('-')
      end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_ADJPRON;

(* sp{cifications : @ partir des donn{es enregistr{es dans ligne, REMPLI_AM_ADJPRON*)
(*                  remplit 'noeud'(= analyse morphologique d'un adj. pronom ).  *)

begin
      case ligne.sscatgram_dgr_vx of
           '1' : noeud.catadjpron := pers;
           '2' : noeud.catadjpron := pos;
           '3' : noeud.catadjpron := refl;
           '4' : noeud.catadjpron := posrefl;
           '5' : noeud.catadjpron := dem;
           '6' : noeud.catadjpron := relat;
           '7' : noeud.catadjpron := inter;
           '8' : noeud.catadjpron := ind;
      end;
      case ligne.cas_pers_nbr of
           'A', 'J' : noeud.casadjpron := nom;
           'B', 'K' : noeud.casadjpron := voc;
           'C', 'L' : noeud.casadjpron := acc;
           'D', 'M' : noeud.casadjpron := gen;
           'E', 'N' : noeud.casadjpron := datif;
           'F', 'O' : noeud.casadjpron := abl;
           'G', 'P' : noeud.casadjpron := loc;
```

```
          'Z'      : begin
                          noeud.casadjpron := indecl;
                          noeud.nbradjpron := nombrenul
                     end
     end;
     case ligne.cas_pers_nbr of
          'A', 'B', 'C', 'D', 'E', 'F', 'G' : noeud.nbradjpron := sing;
          'J', 'K', 'L', 'M', 'N', 'O', 'P' : noeud.nbradjpron := pluriel
     end;
     case ligne.genre of
          '1' : noeud.genreadjpron := commun;
          '2' : noeud.genreadjpron := fem;
          '3' : noeud.genreadjpron := mascfem;
          '4' : noeud.genreadjpron := masc;
          '5' : noeud.genreadjpron := mascneutre;
          '6' : noeud.genreadjpron := neutre;
          else  noeud.genreadjpron := genrenul
     end;
     case ligne.mode of                (* si pronom relatif ou interrogatif *)
          '1' : noeud.modesubadjpron := indic;
          '2' : noeud.modesubadjpron := imp;
          '3' : noeud.modesubadjpron := subj;
          '4' : noeud.modesubadjpron := part;
          '5' : noeud.modesubadjpron := adjverb;
          '6' : noeud.modesubadjpron := gerondif;
          '7' : noeud.modesubadjpron := inf;
          '8' : noeud.modesubadjpron := supinum;
          '9' : noeud.modesubadjpron := supinu;
          else  noeud.modesubadjpron := modenul
     end;
     case ligne.temps of               (* si pronom relatif ou interrogatif *)
          '1' : noeud.tempssubadjpron := pres;
          '2' : noeud.tempssubadjpron := impft;
          '3' : noeud.tempssubadjpron := futspl;
          '4' : noeud.tempssubadjpron := prft;
          '5' : noeud.tempssubadjpron := plusqueprft;
          '6' : noeud.tempssubadjpron := futant;
          '7', '8', '9' : noeud.tempssubadjpron := fuisse;
          else noeud.tempssubadjpron := tempsnul
     end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_ADJPRON;

(* sp{cifications : AFFICH_AM_ADJPRON affiche @ partir de la ligne 'ligne'     *)
(*                  jusqu'@ la ligne 'ligne+3', les informations contenues dans*)
(*                  'noeud' concernant une analyse morphologique d'un adj. prn.*)

begin
     gotoxy (1, ligne);
     write ('Cat{gorie de cet adjectif pronom : ');
     case noeud.catadjpron of
          pers    : write ('personnel');
          pos     : write ('possessif');
          refl    : write ('r{fl{chi');
          posrefl : write ('possessif r{fl{chi');
          dem     : write ('d{monstratif');
          relat   : write ('relatif');
          inter   : write ('interrogatif');
          ind     : write ('ind{fini')
     end;
     gotoxy (1, ligne+1);
     write ('Cas : ');
```

```
      case noeud.casadjpron of
           nom    : write ('nominatif');
           voc    : write ('vocatif');
           acc    : write ('accusatif');
           gen    : write ('g{nitif');
           datif  : write ('datif');
           abl    : write ('ablatif');
           loc    : write ('locatif');
           indecl : write ('ind{clinable')
      end;
      gotoxy (40, ligne+1);
      write ('Nombre : ');
      case noeud.nbradjpron of
           sing      : write ('singulier');
           pluriel   : write ('pluriel');
           nombrenul : write ('-')
      end;
      gotoxy (1, ligne+2);
      write ('Genre : ');
      case noeud.genreadjpron of
           commun     : write ('commun');
           fem        : write ('f{minin');
           mascfem    : write ('masculin-f{minin');
           masc       : write ('masculin');
           mascneutre : write ('masculin-neutre');
           neutre     : write ('neutre');
           genrenul   : write ('-')
      end;
      gotoxy (1, ligne+3);
      write ('Mode Subordonn{ : ');
      case noeud.modesubadjpron of
           indic     : write ('indicatif');
           imp       : write ('imp{ratif');
           subj      : write ('subjonctif');
           part      : write ('participe');
           adjverb   : write ('adjectif verbal');
           gerondif  : write ('g{rondif');
           inf       : write ('infinitif');
           supinum   : write ('supin en ''um''');
           supinu    : write ('supin en ''u''');
           modenul   : write ('-')
      end;
      gotoxy (40, ligne+3);
      write ('Temps Subordonn{ : ');
      case noeud.tempssubadjpron of
           pres         : write ('pr{sent');
           impft        : write ('imparfait');
           futspl       : write ('futur simple');
           prft         : write ('parfait');
           plusqueprft  : write ('plus que parfait');
           futant       : write ('futur ant{rieur');
           fuisse       : write ('''fuisse''');
           tempsnul     : write ('-')
      end
end;

(*------------------------------------------------------------------------*)

procedure REMPLI_AM_CONJ;

(* sp{cifications : @ partir des donn{es enregistr{es dans ligne, REMPLI_AM_CONJ*)
(*                  remplit 'noeud'(= analyse morphologique d'une conjonction). *)

begin
      case ligne.sscatgram_dgr_vx of
```

```
          '1' : noeud.catconj := coord;
          '2' : noeud.catconj := sub
     end;
     case ligne.mode of              (* si conjonction de subordination *)
          '1' : noeud.modesubconj := indic;
          '2' : noeud.modesubconj := imp;
          '3' : noeud.modesubconj := subj;
          '4' : noeud.modesubconj := part;
          '5' : noeud.modesubconj := adjverb;
          '6' : noeud.modesubconj := gerondif;
          '7' : noeud.modesubconj := inf;
          '8' : noeud.modesubconj := supinum;
          '9' : noeud.modesubconj := supinu;
          else  noeud.modesubconj := modenul
     end;
     case ligne.temps of             (* si conjonction de subordination *)
          '1' : noeud.tempssubconj := pres;
          '2' : noeud.tempssubconj := impft;
          '3' : noeud.tempssubconj := futspl;
          '4' : noeud.tempssubconj := prft;
          '5' : noeud.tempssubconj := plusqueprft;
          '6' : noeud.tempssubconj := futant;
          '7', '8', '9' : noeud.tempssubconj := fuisse;
          else noeud.tempssubconj := tempsnul
     end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_CONJ;

(* sp{cifications : AFFICH_AM_CONJ affiche @ partir de la ligne 'ligne'     *)
(*                  jusqu'@ la ligne 'ligne+3', les informations contenues dans*)
(*                  'noeud' concernant une analyse morphologique d'une conj.*)

begin
     gotoxy (1, ligne);
     write ('Cat{gorie de cette conjonction : ');
     case noeud.catconj of
          coord    : write ('coordination');
          sub      : write ('subordination')
     end;
     gotoxy (1, ligne+1);
     write ('Mode Subordonn{ : ');
     case noeud.modesubconj of
          indic     : write ('indicatif');
          imp       : write ('imp{ratif');
          subj      : write ('subjonctif');
          part      : write ('participe');
          adjverb   : write ('adjectif verbal');
          gerondif  : write ('g{rondif');
          inf       : write ('infinitif');
          supinum   : write ('supin en ''um''');
          supinu    : write ('supin en ''u''');
          modenul   : write ('-')
     end;
     gotoxy (1, ligne+2);
     write ('Temps Subordonn{ : ');
     case noeud.tempssubconj of
          pres         : write ('pr{sent');
          impft        : write ('imparfait');
          futspl       : write ('futur simple');
          prft         : write ('parfait');
          plusqueprft  : write ('plus que parfait');
          futant       : write ('futur ant{rieur');
```

```
          fuisse       : write ('''fuisse''');
          tempsnul     : write ('-')
     end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_PREP;

(* sp{cifications : @ partir des donn{es enregistr{es dans ligne, REMPLI_AM_PREP *)
(*                  remplit 'noeud' (= analyse morphologique d'une pr{position). *)

begin
     case ligne.sscatgram_dgr_vx of
          '1' : noeud.typemecum := true;
          else  noeud.typemecum := false
     end;
     case ligne.cas_pers_nbr of
          '3' : noeud.casprep := accusatif;
          '4' : noeud.casprep := genitif;
          '6' : noeud.casprep := ablatif
     end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_PREP;

(* sp{cifications : AFFICH_AM_PREP affiche @ partir de la ligne 'ligne' et *)
(*                  jusqu'@ la ligne 'ligne+1' les informations contenues  *)
(*                  dans 'noeud' concernant une analyse morphologique d'une pr{position*)

begin
     gotoxy (1, ligne);
     write ('Type MECUM ? : ');
     case noeud.typemecum of
          true : write ('oui');
          false : write ('non')
     end;
     gotoxy (1, ligne+1);
     write ('Cas r{gi par cette pr{position : ');
     case noeud.casprep of
          accusatif : write ('accusatif');
          genitif : write ('g{nitif');
          ablatif : write ('ablatif')
     end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICHN3;

(* sp{cifications : sa^chant que 'noeud' est un noeud d'analyse de type n3,   *)
(*                  AFFICHN3 affiche entre les lignes 'miny' et 'maxy',      *)
(*                  les mots correspondant @ 'noeud.ensmots3' @ partir de    *)
(*                  la liste accessible par 'firstmcp'.                      *)

var
    mcpcour : lcontphr;
    i, longreelle, k : integer;
    ch : char;
    stop : boolean;
    motaecrire : str20;

begin
```

```
      clrscr;
      mcpcour := firstmcp;
      i := 1;
      while (mcpcour <> nil) do
      begin
            if (noeud.ensmots3[i] = true)
            then begin
                      if (wherey > maxy)
                      then begin
                                gotoxy (35, maxy + 2);
                                write ('Tapez une touche pour voir la page suivante');
                                repeat until keypressed;
                                ch := readkey;
                                clrscr
                           end;
                      stop := false;
                      k := 1;
                      repeat
                            if (mcpcour^.mot[k] = ' ')
                            then stop := true
                            else k := k + 1
                      until ( (k > 20) or (stop = true) );
                      longreelle := k - 1;
                      motaecrire := copy (mcpcour^.mot, 1, longreelle);
                      if ((80 - wherex) < longreelle) then writeln;
                      write (motaecrire);
                      if (wherex = 1) then gotoxy (80, wherey - 1)
                                      else begin
                                                write (' ');
                                                if (wherex = 1) then gotoxy (80, wherey - 1)
                                           end
                 end;
            i := i + 1;
            mcpcour := mcpcour^.suivant
      end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICHLN3;

(* sp(cifications : AFFICHLN3 affiche entre les lignes 'miny' et 'maxy',     *)
(*                  les mots 'accept(s' appartenant @ la liste accessible    *)
(*                  par 'firstmcp'.                                          *)

var
    ln3cour : lcontphr;
    longreelle, k : integer;
    ch : char;
    stop : boolean;
    motaecrire : str20;

begin
      clrscr;
      ln3cour := firstln3;
      while (ln3cour <> nil) do
      begin
            if (ln3cour^.libre = true)
            then begin
                      if (wherey > maxy)
                      then begin
                                gotoxy (35, maxy + 2);
                                write ('Tapez une touche pour voir la page suivante');
                                repeat until keypressed;
                                ch := readkey;
```

```
                        clrscr
                    end;
                stop := false;
                k := 1;
                repeat
                    if (ln3cour^.mot[k] = ' ')
                    then stop := true
                    else k := k + 1
                until ( (k > 20) or (stop = true) );
                longreelle := k - 1;
                motaecrire := copy (ln3cour^.mot, 1, longreelle);
                if ((80 - wherex) < longreelle) then writeln;
                write (motaecrire);
                if (wherex = 1) then gotoxy (80, wherey - 1)
                        else begin
                                write (' ');
                                if (wherex = 1) then gotoxy (80, wherey - 1)
                             end
            end;
        ln3cour := ln3cour^.suivant
    end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_N3_LN3;

(* sp{cifications : REMPLI_N3_LN3 rempli 'ensmots3' @ partir des informations*)
(*                  enregistr{es dans la liste accessible par 'firstln3',    *)
(*                  et @ partir de la liste de mots accessible par 'firstmcp'*)

var
   ln3cour, mcpcour : lcontphr;
   i : integer;

begin
     ln3cour := firstln3;
     for i := 1 to 85 do ensmots3[i] := false;
     while (ln3cour <> nil) do
     begin
          if (ln3cour^.libre = true)
          then begin
                    mcpcour := firstmcp;
                    i := 1;
                    while (mcpcour^.mot <> ln3cour^.mot) do
                    begin
                         mcpcour := mcpcour^.suivant;
                         i := i + 1
                    end;
                    if (ensmots3[i] = false)
                    then ensmots3[i] := true
                    else begin
                              repeat
                                   begin
                                        mcpcour := mcpcour^.suivant;
                                        i := i + 1
                                   end
                              until ( (mcpcour^.mot = ln3cour^.mot) and (ensmots3[i] = false) );
                              ensmots3[i] := true
                         end
               end;
          ln3cour := ln3cour^.suivant
     end
end;
```

```
(*-------------------------------------------------------------------------*)

procedure TRSF_N3_LN3;

(* sp{cifications : TRSF_N3_LN3 cr{{e la liste accessible par 'firstln3'    *)
(*                  @ partir des donn{es enregistr{es dans 'ensmots3' et de *)
(*                  la liste accessible par 'firstmcp'.                     *)

var
    ln3cour, mcpcour, ln3 : lcontphr;
    i, k : integer;

begin
     ln3cour := nil;
     i := 1;
     for i := 1 to 85 do
     begin
          if (ensmots3[i] = true)       (* s{lectionn{ *)
          then begin
                    mcpcour := firstmcp;
                    k := 1;
                    while (k < i) do
                    begin
                         mcpcour := mcpcour^.suivant;
                         k := k + 1
                    end;
                    new (ln3);
                    ln3^.mot := mcpcour^.mot;
                    ln3^.libre := true;          (* s{lectionn{ *)
                    ln3^.suivant := nil;
                    if (ln3cour = nil)
                    then begin
                              firstln3 := ln3;
                              ln3cour := ln3
                         end
                    else begin
                              ln3cour^.suivant := ln3;
                              ln3cour := ln3cour^.suivant
                          end
               end
     end
end;

(*-------------------------------------------------------------------------*)

procedure MAJLIBRE;

(* sp{cifications : sa^chant que 'noeud' est un noeud d'analyse de type n1,  *)
(*                  MAJLIBRE met @ jour la mention 'libre' dans la liste    *)
(*                  accessible par 'firstmcp' de mani}re @ savoir que ce mot*)
(*                  a {t{ selectionn{ pour une certaine fonction et ne peut *)
(*                  donc plus l'e^tre pour une autre fonction.              *)

var
    mcpcour : lcontphr;
    mememot : str20;

begin
     mcpcour := firstmcp;
     while (mcpcour^.mot <> noeud.mot1)
     do mcpcour := mcpcour^.suivant;
     if (mcpcour^.libre = true)
     then mcpcour^.libre := false
     else begin
               mememot := mcpcour^.mot;
```

```
                repeat
                      mcpcour := mcpcour^.suivant
                until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = true) );
                mcpcour^.libre := false
          end
end;

(*-------------------------------------------------------------------------*)

procedure MAJLNACOUR;

var
    trouve : boolean;
    mcpcour : lcontphr;
    mememot : str20;

begin
     if (lnacour^.noeudfils <> nil) then lnacour := lnacour^.noeudfils;
     fin := false;
     remonte := false;
     trouve := false;
(* on remonte d'office si le noeud est un noeud de traduction ! *)
     if (lnacour^.noeud.typenoeud <> n4)
     then begin
               if (lnacour^.noeud.ind_brk = indicatif)
               then trouve := true
               else while ( (lnacour^.noeudfrere <> nil) and (trouve = false) ) do
                    begin
                         lnacour := lnacour^.noeudfrere;
                         if (lnacour^.noeud.ind_brk = indicatif)
                         then trouve := true
                    end
         end;
     if (trouve = true)
     then if (lnacour^.noeudfils <> nil) then existfna := true
                                         else existfna := false
     else begin
               remonte := true;
               lnacour := lnacour^.noeudpere;
               while ( (lnacour <> nil) and (trouve = false) ) do
               begin
                    while ( (lnacour^.noeudfrere <> nil) and (trouve = false) ) do
                    begin
                         if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.ind_brk = indicatif) )
                         then if ( (lnacour^.noeudpere = nil)
                                or (lnacour^.noeudpere^.noeud.typenoeud = n3)
                                or (lnacour^.noeudpere^.noeudpere^.noeud.fonction = lnacour^.noeud.fonction) )
                              then begin
                                        mcpcour := firstmcp;
                                        while (mcpcour^.mot <> lnacour^.noeud.mot1)
                                        do mcpcour := mcpcour^.suivant;
                                        if (mcpcour ^.libre = false)
                                        then mcpcour^.libre := true
                                        else begin
                                                  mememot := mcpcour^.mot;
                                                  repeat mcpcour := mcpcour^.suivant
                                                  until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = false) );
                                                  mcpcour^.libre := true
                                             end
                                   end;
                         lnacour := lnacour^.noeudfrere;
                         if (lnacour^.noeud.ind_brk = indicatif) then trouve := true
                    end;
                    if (trouve = false)
                    then begin
```

```
                                if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.ind_brk = indicatif) )
                                then if ( (lnacour^.noeudpere = nil)
                                      or (lnacour^.noeudpere^.noeud.typenoeud = n3)
                                      or (lnacour^.noeudpere^.noeudpere^.noeud.fonction = lnacour^.noeud.fonction) )
                                     then begin
                                               mcpcour := firstmcp;
                                               while (mcpcour^.mot <> lnacour^.noeud.mot1)
                                               do mcpcour := mcpcour^.suivant;
                                               if (mcpcour ^.libre = false)
                                               then mcpcour^.libre := true
                                               else begin
                                                         mememot := mcpcour^.mot;
                                                         repeat mcpcour := mcpcour^.suivant
                                                         until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = false)
                                                         mcpcour^.libre := true
                                                    end
                                          end;
                                     lnacour := lnacour^.noeudpere
                                end
                           end;
                           if (trouve = true)
                           then if (lnacour^.noeudfils <> nil) then existfna := true
                                                               else existfna := false
                           else fin := true
                      end
end;


End.
```

```
Unit TRTVP;

Interface
uses ($U b:globald2 ) GLOBALDECL2, CRT, DOS,
    ($U b:lp_unit ) LP_UNIT, ($U b:cpltproc ) CPLTPROC;

procedure TRT_VP ( var existfna : boolean; var firstmcp : lcontphr;
                   var accesfamml : boolean; var lnacour : listnoeudaa;
                   var firstlna : listnoeudaa; var finanbool : boolean;
                   remonte : boolean; var finanal : boolean);

Implementation

procedure TRT_VP;   ( verbe = verbe simple <> "esse" ou "esse" + participe pass{ )

var
    mcpcour : lcontphr;
    position : integer;
    trouve, cheminok, egalite, exist, remontee, fin : boolean;
    newlna, noeudfilscour : listnoeudaa;
    ch : char;

label AFFICH1, AFFICH2, AFFICHPP, AFFICHPP2, ANAL, ANALPP, PART, FINTRTVP;

begin
     finanbool := false;
     finanal := false;
     if (remonte = true)
     then begin
               case lnacour^.noeud.typenoeud of
                    n1 : if (lnacour^.noeudpere = nil) then goto ANAL
                                                       else goto ANALPP;
                    n2 : if (lnacour^.noeudpere^.noeudpere = nil) then goto PART
                                                                  else goto FINTRTVP
               end
          end;
     if (existfna = true)
     then goto AFFICH1  ( prise en compte de l'existence de l'analyse )
     else begin
               ( recherche des candidats verbes principaux )
               mcpcour := firstmcp;
               assign (famml, 'b:fmotlat.am');
               reset (famml);
               accesfamml := true;
               lnacour := nil;
               while (mcpcour <> nil) do
               begin
                    if (mcpcour^.libre = true)
                    then begin
                              FIND_INDEX (mcpcour^.mot, position);
                              seek (famml, position);
                              trouve := false;
                              while ( (not eof(famml)) and (trouve = false) ) do
                              begin
                                   read (famml, lignefamml);
                                   if (lignefamml.motlat <> mcpcour^.mot)
                                   then trouve := true
                                   else if ( ((lignefamml.catgram = '5') or (lignefamml.catgram = 'E'))
                                             and ((lignefamml.mode = '1') or (lignefamml.mode = '2')
                                                  or (lignefamml.mode = '3') or (lignefamml.mode = '7')) )
                                        then begin
                                                  trouve := true;
                                                  new (newlna);
                                                  newlna^.noeudfils := nil;
                                                  newlna^.noeudfrere := nil;
```

```
                                                newlna^.noeudpere := nil;
                                                newlna^.noeud.appart_analyse_correct := true;
                                                newlna^.noeud.typenoeud := n1;
                                                newlna^.noeud.fonction := verbe;
                                                newlna^.noeud.mot1 := lignefamml.motlat;
                                                newlna^.noeud.comment_syst := '-';
                                                if (lnacour = nil)
                                                then begin
                                                          lnacour := newlna;
                                                          firstlna := newlna
                                                     end
                                                else begin
                                                          lnacour^.noeudfrere := newlna;
                                                          lnacour := lnacour^.noeudfrere
                                                     end
                                           end;
                                 end
                            end;
                       mcpcour := mcpcour^.suivant
                  end
             end;

AFFICH1 :
(* ayant ces noeuds comprenant les verbes principaux possibles, il faut les *)
(* proposer @ l'enseignant pour qu'il donne ses renseignements suppl{mentaires *)
     lnacour := firstlna;
     cheminok := false;
     while (lnacour <> nil) do
     begin
          clrscr;
          gotoxy (1, 2);
          write ('Candidat Verbe Principal : ');
          write (lnacour^.noeud.mot1);
          gotoxy (1, 4);
          writeln ('Commentaire du syst}me :');
          write (lnacour^.noeud.comment_syst);
          AFFCOMM (7, lnacour^.noeud.comment_specif, existfna);
          AFFACCCOMM (10, lnacour^.noeud.accept_comment, existfna);
          AFFINDBRK (12, lnacour, existfna, cheminok);
          gotoxy (40, 14);
          write ('Tapez une touche pour continuer ou ESC !');
          repeat until keypressed;
          ch := readkey;
          if (ch = #27) then begin
                                 finanbool := true;
                                 goto FINTRTVP
                            end;
          lnacour := lnacour^.noeudfrere
     end;
(* analyse morphologique du 1[ candidat 'verbe principal' accept{ :          *)
(* recherche automatique des analyses morphologiques verbales possibles ou   *)
(* acc}s @ l'analyse d{j@ existante, et propositions @ l'enseignant pour compl{ments. *)
     lnacour := firstlna;
     trouve := false;
     while ((lnacour <> nil) and (trouve = false)) do
     begin
          if (lnacour^.noeud.ind_brk = indicatif)
          then trouve := true
          else lnacour := lnacour^.noeudfrere
     end;
     if (trouve = true)
     then begin
               if (lnacour^.noeudfils <> nil) then existfna := true
                                              else existfna := false
          end
```

```
     else begin
               finanbool := true;
               finanal := true;
               goto FINTRTVP
          end;
ANAL :
     MAJLIBRE (firstmcp, lnacour^.noeud);
     if (existfna = false)
     then begin
               noeudfilscour := lnacour^.noeudfils;
(* recherche dans le fichier INDEX de la position de la 1[ analyse morphologique *)
(* possible de ce mot, puis recherche dans le fichier des analyses morphologiques*)
(* et s{lection des analyses verbales .                                          *)
               if (accesfamml = false) then begin
                                              assign (famml, 'b:fmotlat.am');
                                              reset (famml);
                                              accesfamml := true
                                         end;
               FIND_INDEX (lnacour^.noeud.mot1, position);
               seek (famml, position);
               egalite := true;
               while ( (not eof(famml)) and (egalite = true) ) do
               begin
                    read (famml, lignefamml);
                    if (lignefamml.motlat <> lnacour^.noeud.mot1)
                    then egalite := false
                    else if ( ((lignefamml.catgram = '5') or (lignefamml.catgram = 'E'))
                         and ((lignefamml.mode = '1') or (lignefamml.mode = '2')
                           or (lignefamml.mode = '3') or (lignefamml.mode = '7')))
                        then begin
                                  new (newlna);
                                  newlna^.noeudfils := nil;
                                  newlna^.noeudfrere := nil;
                                  newlna^.noeudpere := lnacour;
                                  newlna^.noeud.appart_analyse_correct := true;
                                  newlna^.noeud.typenoeud := n2;
                                  newlna^.noeud.lemme := lignefamml.lemme;
                                  newlna^.noeud.categorie := v;
                                  newlna^.noeud.comment_syst := concat ('lemme : ', lignefamml.lemme);
                                  REMPLI_AM_V (newlna^.noeud, lignefamml);
                                  if (noeudfilscour = nil)
                                  then begin
                                            lnacour^.noeudfils := newlna;
                                            noeudfilscour := lnacour^.noeudfils
                                       end
                                  else begin
                                            noeudfilscour^.noeudfrere := newlna;
                                            noeudfilscour := noeudfilscour^.noeudfrere
                                       end
                             end
               end
          end;
AFFICH2 :
     (* pr{sentation des r{sultats @ l'enseignant pour compl{ments .          *)
     noeudfilscour := lnacour^.noeudfils;
     cheminok := false;
     while (noeudfilscour <> nil) do
     begin
          clrscr;
          gotoxy (1, 2);
          write ('Analyse du Candidat Verbe Principal Possible : ');
          write (lnacour^.noeud.mot1);
          AFFICH_AM_V (4, noeudfilscour^.noeud);
          gotoxy (1, 9);
          writeln ('Commentaire du syst}me :');
```

```
          write (noeudfilscour^.noeud.comment_syst);
          AFFCOMM (11, noeudfilscour^.noeud.comment_specif, existfna);
          AFFACCCOMM (13, noeudfilscour^.noeud.accept_comment, existfna);
          AFFINDBRK (15, noeudfilscour, existfna, cheminok);
          gotoxy (40, 15);
          write ('Tapez une touche pour continuer ou ESC !');
          repeat until keypressed;
          ch := readkey;
          if (ch = #27) then begin
                                finanbool := true;
                                goto FINTRTVP
                            end;
          noeudfilscour := noeudfilscour^.noeudfrere
     end;
(* prendre la premi}re analyse morphologique accept{e et continuer l'analyse *)
(* si lemme = 'SVM' alors proposer les participes pass{s possibles,  etc...  *)
(* toujours consid{rer 'si existfna = true'                                   *)
(* mettre @ jour 'mcpcour^.libre' de mani}re @ limiter les mots de la phrase *)
(* non encore trait{s (descente = limite, remont{e = rela^che).               *)
     MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
     if (remontee = true)
     then if (fin = true)
          then begin
                    finanbool := true;
                    finanal := true;
                    goto FINTRTVP
               end
          else goto ANAL;
PART :
     if (lnacour^.noeud.lemme = 'SVM            ')
     then begin
               if (existfna = true)
               then goto AFFICHPP
               else begin
                         (* recherche des participes pass{s *)
                         mcpcour := firstmcp;
                         if (accesfamml = false) then begin
                                                        assign (famml, 'b:fmotlat.am');
                                                        reset (famml);
                                                        accesfamml := true
                                                      end;
                         noeudfilscour := lnacour^.noeudfils;
                         exist := false;
                         while (mcpcour <> nil) do
                         begin
                              if (mcpcour^.libre = true)
                              then begin
                                        FIND_INDEX (mcpcour^.mot, position);
                                        seek (famml, position);
                                        trouve := false;
                                        while ( (not eof(famml)) and (trouve = false) ) do
                                        begin
                                             read (famml, lignefamml);
                                             if (lignefamml.motlat <> mcpcour^.mot)
                                             then trouve := true
                                             else if ((lignefamml.catgram = '5') and (lignefamml.mode = '4'))
                                                  then begin
                                                            trouve := true;
                                                            exist := true;
                                                            new (newlna);
                                                            newlna^.noeudfils := nil;
                                                            newlna^.noeudfrere := nil;
                                                            newlna^.noeudpere := lnacour;
                                                            newlna^.noeud.appart_analyse_correct := true;
                                                            newlna^.noeud.typenoeud := n1;
```

```
                                                    newlna^.noeud.fonction := verbe;
                                                    newlna^.noeud.mot1 := lignefamml.motlat;
                                                    newlna^.noeud.comment_syst := '-';
                                                    if (noeudfilscour = nil)
                                                    then begin
                                                              lnacour^.noeudfils := newlna;
                                                              noeudfilscour := newlna
                                                         end
                                                    else begin
                                                              noeudfilscour^.noeudfrere := newlna;
                                                              noeudfilscour := noeudfilscour^.noeudfrere
                                                         end
                                               end;
                                          end
                                     end;
                                mcpcour := mcpcour^.suivant
                           end
                      end;
                (* affichage des donn{es concernant les participes pass{s trouv{s *)
                if (exist = true)
                then
                AFFICHPP :
                     begin
                          noeudfilscour := lnacour^.noeudfils;
                          cheminok := false;
                          while (noeudfilscour <> nil) do
                          begin
                               clrscr;
                               gotoxy (1, 2);
                               write ('Candidat Participe Pass{ associ{ @ ', lnacour^.noeudpere^.noeud.mot1, (' : '));
                               write (noeudfilscour^.noeud.mot1);
                               gotoxy (1, 4);
                               writeln ('Commentaire du syst}me : ');
                               write (noeudfilscour^.noeud.comment_syst);
                               AFFCOMM (7, noeudfilscour^.noeud.comment_specif, existfna);
                               AFFACCCOMM (10, noeudfilscour^.noeud.accept_comment, existfna);
                               AFFINDBRK (13, noeudfilscour, existfna, cheminok);
                               gotoxy (40, 14);
                               write ('Tapez une touche pour continuer ou ESC !');
                               repeat until keypressed;
                               ch := readkey;
                               if (ch = #27) then begin
                                                       finanbool := true;
                                                       goto FINTRTVP
                                                  end;
                               noeudfilscour := noeudfilscour^.noeudfrere
                          end;
                          (* analyses morphologiques du 1[ part. pass{ *)
                          MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
                          if (remontee = true)
                          then if (fin = true)
                               then begin
                                         finanbool := true;
                                         finanal := true;
                                         goto FINTRTVP
                                    end
                               else begin
                                         if (lnacour^.noeud.typenoeud = n1)
                                         then goto ANAL
                                         else goto PART
                                    end;
ANALPP:
                          MAJLIBRE (firstmcp, lnacour^.noeud);
                          if (existfna = false)
                          then begin
```

```
                    if (accesfamml = false) then begin
                                                    assign (famml, 'b:fmotlat.am');
                                                    reset (famml);
                                                    accesfamml := true
                                                end;
                    noeudfilscour := lnacour^.noeudfils;
                    FIND_INDEX (lnacour^.noeud.mot1, position);
                    seek (famml, position);
                    egalite := true;
                    while ( (not eof(famml)) and (egalite = true) ) do
                    begin
                         read (famml, lignefamml);
                         if (lignefamml.motlat <> lnacour^.noeud.mot1)
                         then egalite := false
                         else if ((lignefamml.catgram = '5') and (lignefamml.mode = '4'))
                         then begin
                                   new (newlna);
                                   newlna^.noeudfils := nil;
                                   newlna^.noeudfrere := nil;
                                   newlna^.noeudpere := lnacour;
                                   newlna^.noeud.appart_analyse_correct := true;
                                   newlna^.noeud.typenoeud := n2;
                                   newlna^.noeud.lemme := lignefamml.lemme;
                                   newlna^.noeud.categorie := v;
                                   newlna^.noeud.comment_syst:=concat('lemme : ',lignefamml.lemme);
                                   REMPLI_AM_V (newlna^.noeud, lignefamml);
                                   if (noeudfilscour = nil)
                                   then begin
                                             lnacour^.noeudfils := newlna;
                                             noeudfilscour := lnacour^.noeudfils
                                        end
                                   else begin
                                             noeudfilscour^.noeudfrere := newlna;
                                             noeudfilscour := noeudfilscour^.noeudfrere
                                        end
                              end
                    end
               end;

          AFFICHPP2 :
(* pr{sentation des r{sultats @ l'enseignant pour compl{ments .          *)
          noeudfilscour := lnacour^.noeudfils;
          cheminok := false;
          while (noeudfilscour <> nil) do
          begin
               clrscr;
               gotoxy (1, 2);
               write ('Analyse du Candidat Participe Pass{ Possible : ');
               write (lnacour^.noeud.mot1);
               AFFICH_AM_V (4, noeudfilscour^.noeud);
               gotoxy (1, 9);
               writeln ('Commentaire du syst}me :');
               write (noeudfilscour^.noeud.comment_syst);
               AFFCOMM (11, noeudfilscour^.noeud.comment_specif, existfna);
               AFFACCCOMM (13, noeudfilscour^.noeud.accept_comment, existfna);
               AFFINDBRK (15, noeudfilscour, existfna, cheminok);
               gotoxy (40, 15);
               write ('Tapez une touche pour continuer ou ESC !');
               repeat until keypressed;
               ch := readkey;
               if (ch = #27) then begin
                                       finanbool := true;
                                       goto FINTRTVP
                                  end;
               noeudfilscour := noeudfilscour^.noeudfrere
```

```
                    end;
                    MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
                    if (remontee = true)
                    then if (fin = true)
                         then begin
                                   finanbool := true;
                                   finanal := true;
                                   goto FINTRTVP
                              end
                         else begin
                                   if (lnacour^.noeud.typenoeud = n2)
                                   then goto PART
                                   else if (lnacour^.noeudpere <> nil)
                                        then goto ANALPP
                                        else goto ANAL
                                end;
               end
     end;
FINTRTVP :
    write ('')
end;

End.
```

```
Unit TRTGN;

Interface
uses ('$U b:globald2 ') GLOBALDECL2, CRT, DOS, ('$U b:lp_unit ') LP_UNIT,
    ('$U b:cpltproc ') CPLTPROC;

procedure TRT_GN (var existfna : boolean; var firstmcp : lcontphr; var accesfamml : boolean;
                  var lnacour : listnoeudaa; var finanbool : boolean; var noeudmem : listnoeudaa;
                  casgn : cas; remonte : boolean; var trtsuivant : traitement; var finanal : boolean);

Implementation

procedure TRT_GN;

(* traitement GN : choix 'centre-gn', analyses morphologiques et d(coupe     *)
(* syntaxique suivie de la s(lection et des analyses de chaque mot.          *)

var
    cheminok, ok, accept, egalite, trouve, stop, modif, ajout, remontee, fin, gotoanalgn : boolean;
    mcpcour, ln3, ln3cour, firstln3, ln3temp : lcontphr;
    noeudfilscour, newlna, lnatemp, noeudsuivant : listnoeudaa;
    position, k, longreelle, i : integer;
    motaecrire, mememot : str20;
    cass, casp, casprep, ch : char;

label  AFFICH1, AFFICH2, FINTRTGN, DEBUT, AFFICH, ANAL, CHOIXGN, ANALGN;

begin
     finanbool := false;
     finanal := false;
     noeudmem := nil;
     trtsuivant := trtnul;
     gotoanalgn := false;
     if (remonte = true)
     then begin
               case lnacour^.noeud.typenoeud of
                    n1 : case casgn of
                              nom : if (lnacour^.noeud.fonction = sujet)
                                    then goto ANAL
                                    else begin
                                              trtsuivant := trtv;
                                              goto FINTRTGN
                                         end;
                              acc : if (lnacour^.noeud.fonction = cod)
                                    then goto ANAL
                                    else begin
                                              trtsuivant := trtgns;
                                              goto FINTRTGN
                                         end;
                              abl : if (lnacour^.noeud.fonction = cplt)
                                    then goto ANAL
                                    else begin
                                              trtsuivant := trtgncod;
                                              goto FINTRTGN
                                         end
                         end;
                    n2 : case casgn of
                              nom : if ( ((lnacour^.noeudpere^.noeudpere^.noeud.typenoeud = n3)
                                     and (lnacour^.noeudpere^.noeudpere^.noeud.symbcatsynt = gnsujet))
                                     or (lnacour^.noeudpere^.noeudpere^.noeudpere^.noeud.fonction = sujet) )
                                    then begin
                                              lnatemp := lnacour;
                                              while (lnatemp^.noeud.typenoeud <> n3)
                                              do lnatemp := lnatemp^.noeudpere;
                                              TRSF_N3_LN3 (firstln3, lnatemp^.noeud.ensmots3, firstmcp);
```

```
                                        ln3cour := firstln3;
                                        while (ln3cour^.mot <> lnacour^.noeudpere^.noeud.mot1)
                                        do ln3cour := ln3cour^.suivant;
                                        ln3cour := ln3cour^.suivant;
                                        if (ln3cour <> nil) then begin goto ANALGN; gotoanalgn := true end
                                                            else goto FINTRTGN
                                    end
                                else if (lnacour^.noeudpere^.noeud.fonction <> sujet)
                                    then begin
                                            trtsuivant := trtv;
                                            goto FINTRTGN
                                        end
                                    else goto CHOIXGN;
                            acc : if ( ((lnacour^.noeudpere^.noeudpere^.noeud.typenoeud = n3)
                                  and (lnacour^.noeudpere^.noeudpere^.noeud.symbcatsynt = gncod))
                                  or (lnacour^.noeudpere^.noeudpere^.noeudpere^.noeud.fonction = cod) )
                                then begin
                                        lnatemp := lnacour;
                                        while (lnatemp^.noeud.typenoeud <> n3)
                                        do lnatemp := lnatemp^.noeudpere;
                                        TRSF_N3_LN3 (firstln3, lnatemp^.noeud.ensmots3, firstmcp);
                                        ln3cour := firstln3;
                                        while (ln3cour^.mot <> lnacour^.noeudpere^.noeud.mot1)
                                        do ln3cour := ln3cour^.suivant;
                                        ln3cour := ln3cour^.suivant;
                                        if (ln3cour <> nil) then begin goto ANALGN; gotoanalgn := true end
                                                            else goto FINTRTGN
                                    end
                                else if (lnacour^.noeudpere^.noeud.fonction <> cod)
                                    then begin
                                            trtsuivant := trtgns;
                                            goto FINTRTGN
                                        end
                                    else goto CHOIXGN;
                            abl : if ( ((lnacour^.noeudpere^.noeudpere^.noeud.typenoeud = n3)
                                  and (lnacour^.noeudpere^.noeudpere^.noeud.symbcatsynt = gnabl))
                                  or (lnacour^.noeudpere^.noeudpere^.noeudpere^.noeud.fonction = cplt) )
                                then begin
                                        lnatemp := lnacour;
                                        while (lnatemp^.noeud.typenoeud <> n3)
                                        do lnatemp := lnatemp^.noeudpere;
                                        TRSF_N3_LN3 (firstln3, lnatemp^.noeud.ensmots3, firstmcp);
                                        ln3cour := firstln3;
                                        while (ln3cour^.mot <> lnacour^.noeudpere^.noeud.mot1)
                                        do ln3cour := ln3cour^.suivant;
                                        ln3cour := ln3cour^.suivant;
                                        if (ln3cour <> nil) then begin goto ANALGN; gotoanalgn := true end
                                                            else goto FINTRTGN
                                    end
                                else if (lnacour^.noeudpere^.noeud.fonction <> cplt)
                                    then begin
                                            trtsuivant := trtgncod;
                                            goto FINTRTGN
                                        end
                                    else goto CHOIXGN
                        end
                end
        end;
if (existfna = true)
then goto AFFICH1
else begin
        (* recherche de l'{l{ment principal du GN *)
        mcpcour := firstmcp;
        if (accesfamml = false) then begin
                                        assign (famml, 'b:fmotlat.am');
```

```
                                        reset (famml);
                                        accesfamml := true
                                    end;
                noeudfilscour := lnacour^.noeudfils;
                while (mcpcour <> nil) do
                begin
                     if (mcpcour^.libre = true)
                     then begin
                               FIND_INDEX (mcpcour^.mot, position);
                               seek (famml, position);
                               trouve := false;
                               while ( (not eof(famml)) and (trouve = false) ) do
                               begin
                                    read (famml, lignefamml);
                                    if (lignefamml.motlat <> mcpcour^.mot)
                                    then trouve := true
                                    else begin
                                         case casgn of
                                              nom : begin cass := 'A'; casp := 'J' end;
                                              acc : begin cass := 'C'; casp := 'L' end;
                                              abl : begin cass := 'F'; casp := 'O' end
                                         end;
                                         if ( ((lignefamml.catgram = '1') and ((lignefamml.cas_pers_nbr = cass)
                                                                               or (lignefamml.cas_pers_nbr = casp)))
                                            or ((lignefamml.catgram = '4') and ((lignefamml.cas_pers_nbr = cass)
                                                                               or (lignefamml.cas_pers_nbr = casp))
                                                                           and ((lignefamml.sscatgram_dgr_vx = '1')
                                                                                or (lignefamml.sscatgram_dgr_vx = '3')
                                                                                or (lignefamml.sscatgram_dgr_vx = '5')
                                                                                or (lignefamml.sscatgram_dgr_vx = '7')
                                                                                or (lignefamml.sscatgram_dgr_vx = '8')
(* substantif ('1'), singulier ou pluriel, au cas d{sir{ : cass ou casp                                              *)
(* ou pronom ('4'), personnel ('1'), r{fl{chi ('3'), d{monstratif ('5'), interrogatif ('7') ou ind{fini ('8'). *)
                                         then begin
                                                   trouve := true;
                                                   new (newlna);
                                                   newlna^.noeudfils := nil;
                                                   newlna^.noeudfrere := nil;
                                                   newlna^.noeudpere := lnacour;
                                                   newlna^.noeud.appart_analyse_correct := true;
                                                   newlna^.noeud.typenoeud := n1;
                                                   newlna^.noeud.mot1 := lignefamml.motlat;
                                                   case casgn of
                                                        nom : newlna^.noeud.fonction := sujet;
                                                        acc : newlna^.noeud.fonction := cod;
                                                        abl : newlna^.noeud.fonction := cplt
                                                   end;
                                                   newlna^.noeud.comment_syst := '-';
                                                   if (noeudfilscour = nil)
                                                   then begin
                                                             lnacour^.noeudfils := newlna;
                                                             noeudfilscour := newlna
                                                        end
                                                   else begin
                                                             noeudfilscour^.noeudfrere := newlna;
                                                             noeudfilscour := noeudfilscour^.noeudfrere
                                                        end
                                              end
                                         end
                               end
                          end;
                     mcpcour := mcpcour^.suivant
                end
          end;
(* affichage des donn{es concernant les candidats {l{ments centraux trouv{s *)
```

```
AFFICH1 :
     noeudfilscour := lnacour^.noeudfils;
     cheminok := false;
     while (noeudfilscour <> nil) do
     begin
          clrscr;
          gotoxy (1, 2);
          case casgn of
               nom : write ('Candidat Sujet : ');
               acc : write ('Candidat COD ''central'' : ');
               abl : write ('Candidat Compl{ment ''central'' : ')
          end;
          write (noeudfilscour^.noeud.mot1);
          gotoxy (1, 4);
          writeln ('Commentaire du syst)me : ');
          write (noeudfilscour^.noeud.comment_syst);
          AFFCOMM (7, noeudfilscour^.noeud.comment_specif, existfna);
          AFFACCCOMM (10, noeudfilscour^.noeud.accept_comment, existfna);
          AFFINDBRK (13, noeudfilscour, existfna, cheminok);
          gotoxy (40, 14);
          write ('Tapez une touche pour continuer ou ESC !');
          repeat until keypressed;
          ch := readkey;
          if (ch = #27) then begin
                                finanbool := true;
                                goto FINTRTGN
                            end;
          noeudfilscour := noeudfilscour^.noeudfrere
     end;
(* pas MAJLIBRE @ ce moment, mais pour tout mot du GN *)

(* analyses morphologiques du 1[ {l{ment central accept{ *)
     MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
     if (remontee = true)
     then if (fin = true)
          then begin
                    finanbool := true;
                    finanal := true;
                    goto FINTRTGN
               end
          else begin
                    case casgn of
                         nom : trtsuivant := trtv;
                         acc : trtsuivant := trtgns;
                         abl : trtsuivant := trtgncod
                    end;
                    goto FINTRTGN
               end;
ANAL :
     if (existfna = false)
     then begin
               noeudfilscour := lnacour^.noeudfils;
               if (accesfamml = false) then begin
                                                 assign (famml, 'b:fmotlat.an');
                                                 reset (famml);
                                                 accesfamml := true
                                            end;
               FIND_INDEX (lnacour^.noeud.mot1, position);
               seek (famml, position);
               egalite := true;
               while ( (not eof(famml)) and (egalite = true) ) do
               begin
                    read (famml, lignefamml);
                    if (lignefamml.motlat <> lnacour^.noeud.mot1)
                    then egalite := false
```

```
                    else begin
                              case casgn of
                                   nom : begin cass := 'A'; casp := 'J' end;
                                   acc : begin cass := 'C'; casp := 'L' end;
                                   abl : begin cass := 'F'; casp := 'O' end
                              end;
                              if ( ((lignefamml.catgram = '1') and ((lignefamml.cas_pers_nbr = cass)
                                                                    or (lignefamml.cas_pers_nbr = casp)))
                                or ((lignefamml.catgram = '4') and ((lignefamml.cas_pers_nbr = cass)
                                                                    or (lignefamml.cas_pers_nbr = casp))
                                                                   and ((lignefamml.sscatgram_dgr_vx = '1')
                                                                    or (lignefamml.sscatgram_dgr_vx = '3')
                                                                    or (lignefamml.sscatgram_dgr_vx = '5')
                                                                    or (lignefamml.sscatgram_dgr_vx = '7')
                                                                    or (lignefamml.sscatgram_dgr_vx = '8'))) )
                              then begin
                                        new (newlna);
                                        newlna^.noeudfils := nil;
                                        newlna^.noeudfrere := nil;
                                        newlna^.noeudpere := lnacour;
                                        newlna^.noeud.appart_analyse_correct := true;
                                        newlna^.noeud.typenoeud := n2;
                                        newlna^.noeud.lemme := lignefamml.lemme;
                                        if (lignefamml.catgram = '1')
                                        then begin
                                                  newlna^.noeud.categorie := subst;
                                                  REMPLI_AM_SUBST(newlna^.noeud, lignefamml)
                                             end
                                        else begin newlna^.noeud.categorie := adjpron;
                                                  REMPLI_AM_ADJPRON (newlna^.noeud, lignefamml)
                                             end;
                                        newlna^.noeud.comment_syst:=concat('lemme : ',lignefamml.lemme);
                                        if (noeudfilscour = nil)
                                        then begin
                                                  lnacour^.noeudfils := newlna;
                                                  noeudfilscour := lnacour^.noeudfils
                                             end
                                        else begin
                                                  noeudfilscour^.noeudfrere := newlna;
                                                  noeudfilscour := noeudfilscour^.noeudfrere
                                             end
                                   end
                         end
               end
          end;
AFFICH2 :
     (* pr(sentation des r(sultats @ l'enseignant pour compl(ments .        *)
     noeudfilscour := lnacour^.noeudfils;
     cheminok := false;
     while (noeudfilscour <> nil) do
     begin
          clrscr;
          gotoxy (1, 2);
          case casgn of
               nom : write ('Analyse du Candidat Sujet Possible : ');
               acc : write ('Analyse du Candidat COD ''central'' Possible : ');
               abl : write ('Analyse du Candidat Compl(ment ''central'' Possible : ')
          end;
          write (lnacour^.noeud.mot1);
          if (noeudfilscour^.noeud.categorie = subst)
          then AFFICH_AM_SUBST (4, noeudfilscour^.noeud)
          else AFFICH_AM_ADJPRON (4, noeudfilscour^.noeud);
          gotoxy (1, 9);
          writeln ('Commentaire du syst)me :');
          write (noeudfilscour^.noeud.comment_syst);
```

```
          AFFCOMM (11, noeudfilscour^.noeud.comment_specif, existfna);
          AFFACCCOMM (13, noeudfilscour^.noeud.accept_comment, existfna);
          AFFINDBRK (15, noeudfilscour, existfna, cheminok);
          gotoxy (40, 15);
          write ('Tapez une touche pour continuer ou ESC !');
          repeat until keypressed;
          ch := readkey;
          if (ch = #27) then begin
                                finanbool := true;
                                goto FINTRTGN
                            end;
          noeudfilscour := noeudfilscour^.noeudfrere
     end;
(* s{lection mots appartenant @ ce GN et analyses morphologiques .          *)
     MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
     if (remontee = true)
     then if (fin = true)
          then begin
                    finanbool := true;
                    finanal := true;
                    goto FINTRTGN
               end
          else case casgn of
                    nom : if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.fonction = sujet) )
                          then goto ANAL
                          else begin
                                    trtsuivant := trtv;
                                    goto FINTRTGN
                               end;
                    acc : if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.fonction = cod) )
                          then goto ANAL
                          else begin
                                    trtsuivant := trtgns;
                                    goto FINTRTGN
                               end;
                    abl : if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.fonction = cplt) )
                          then goto ANAL
                          else begin
                                    trtsuivant := trtgncod;
                                    goto FINTRTGN
                               end
               end;
CHOIXGN :
     if (existfna = true)
     then begin
               noeudfilscour := lnacour^.noeudfils;
               AFFICHN3 (noeudfilscour^.noeud, firstmcp, 1, 8);
          end
     else begin
               mcpcour := firstmcp;
               if (accesfamml = false) then begin
                                                 assign (famml, 'b:fmotlat.am');
                                                 reset (famml);
                                                 accesfamml := true
                                            end;
               ln3cour := nil;
               while (mcpcour <> nil) do
               begin
                    if (mcpcour^.libre = true)
                    then begin
                              FIND_INDEX (mcpcour^.mot, position);
                              seek (famml, position);
                              trouve := false;
                              while ((not eof(famml)) and (trouve = false)) do
                              begin
```

```
                                read (famml, lignefamml);
                                if (lignefamml.motlat <> mcpcour^.mot)
                                then trouve := true
                                else begin
                                    case casgn of
                                        nom : begin cass := 'A'; casp := 'J'; casprep := '1' end;
                                        acc : begin cass := 'C'; casp := 'L'; casprep := '3' end;
                                        abl : begin cass := 'F'; casp := 'O'; casprep := '6' end
                                    end;
                                    if ( ( (lignefamml.catgram = '1') and ((lignefamml.cas_pers_nbr = cass)
                                                                    or (lignefamml.cas_pers_nbr = casp)) )
                                      or ( (lignefamml.catgram = '2') and ((lignefamml.cas_pers_nbr = cass)
                                                                    or (lignefamml.cas_pers_nbr = casp)) )
                                      or ( (lignefamml.catgram = '4') and ((lignefamml.cas_pers_nbr = cass)
                                                                    or (lignefamml.cas_pers_nbr = casp)) )
                                      or ( (lignefamml.catgram = '8') and (lignefamml.sscatgram_dgr_vx = '1') )
                                      or ( (lignefamml.catgram = '7') and (lignefamml.cas_pers_nbr = casprep) ) )
                                    then begin
                                            trouve := true;
                                            new (ln3);
                                            ln3^.mot := lignefamml.motlat;
                                            ln3^.libre := true;
                                            ln3^.suivant := nil;
                                            if (ln3cour = nil)
                                            then begin
                                                    firstln3 := ln3;
                                                    ln3cour := ln3
                                                end
                                            else begin
                                                    ln3cour^.suivant := ln3;
                                                    ln3cour := ln3cour^.suivant
                                                end
                                        end
                                    end
                                end;
                            mcpcour := mcpcour^.suivant
                        end;
                        AFFICHLN3 (firstln3, 1, 8)
                    end;
                accept := false;
                modif := false;
                while (accept = false) do
                begin
                    gotoxy (1, 10);
                    case casgn of
                        nom : write ('Acceptez-vous ce GNsujet ? ''O''-''N'' : _');
                        acc : write ('Acceptez-vous ce GNCOD ? ''O''-''N'' : _');
                        abl : write ('Acceptez-vous ce GNcompl(ment ? ''O''-''N'' : _')
                    end;
                    gotoxy (wherex - 1, wherey);
                    ok := false;
                    while (ok = false) do
                    begin
                        if keypressed then begin
                                            ch := readkey;
                                            case ch of
                                                'N', 'n', 'O', 'o' : begin
                                                                        write (ch);
                                                                        ok := true
                                                                    end;
                                                else                begin
                                                                        sound (220);
                                                                        delay (200);
                                                                        nosound
```

```
                                                    end
                                    end
                                end
        end;
        if ((ch = 'o') or (ch = 'O'))
        then begin
                accept := true;
                if (existfna = false)
                then begin
                        new (newlna);
                        newlna^.noeudfils := nil;
                        newlna^.noeudfrere := nil;
                        newlna^.noeudpere := lnacour;
                        newlna^.noeud.appart_analyse_correct := true;
                        newlna^.noeud.typenoeud := n3;
                        case casgn of
                             nom : begin
newlna^.noeud.comment_syst := 'Les {l{ments appartenant au GNsujet sont des nominatifs, associ{s au verbe';
newlna^.noeud.symbcatsynt := gnsujet
                                        end;
                                   acc : begin
newlna^.noeud.comment_syst := 'Les {l{ments appartenant au GNcod sont des accusatifs';
newlna^.noeud.symbcatsynt := gncod
                                        end;
                                   abl : begin
newlna^.noeud.comment_syst := 'Les {l{ments appartenant au GNcompl{ment sont des ablatifs';
newlna^.noeud.symbcatsynt := gnabl
                                        end
                        end;
                        REMPLI_N3_LN3 (firstln3, newlna^.noeud.ensmots3, firstmcp);
                        lnacour^.noeudfils := newlna;
                        noeudfilscour := lnacour^.noeudfils
                    end
                else if (modif = true) then REMPLI_N3_LN3 (firstln3, noeudfilscour^.noeud.ensmots3, firstmcp);
                cheminok := false;
                gotoxy (1, 10);
                delline;
                writeln ('Commentaire du syst}me :');
                write (noeudfilscour^.noeud.comment_syst);
                AFFCOMM (12, noeudfilscour^.noeud.comment_specif, existfna);
                AFFACCCOMM (14, noeudfilscour^.noeud.accept_comment, existfna);
                AFFINDBRK (15, noeudfilscour, existfna, cheminok);
                gotoxy (40, 15);
                write ('Tapez une touche pour continuer ou ESC');
                repeat until keypressed;
                ch := readkey;
                if (ch = #27) then begin
                                        finanbool := true;
                                        goto FINTRTGN
                                   end
            end
        else begin
                 if (existfna = true)
                 then begin
                        mcpcour := firstmcp;
                        modif := true;
                        if (accesfamml = false) then begin
                                                        assign (famml, 'b:fmotlat.am');
                                                        reset (famml);
                                                        accesfamml := true
                                                     end;
                        ln3cour := nil;
                        while (mcpcour <> nil) do
                        begin
                             if (mcpcour^.libre = true)
```

```
                then begin
                        FIND_INDEX (mcpcour^.mot, position);
                        seek (famml, position);
                        trouve := false;
                        while ((not eof(famml)) and (trouve = false)) do
                        begin
                            read (famml, lignefamml);
                            if (lignefamml.motlat <> mcpcour^.mot)
                            then trouve := true
                            else begin
                                case casgn of
                                    nom : begin cass := 'A'; casp := 'J'; casprep := '1' end;
                                    acc : begin cass := 'C'; casp := 'L'; casprep := '3' end;
                                    abl : begin cass := 'F'; casp := 'O'; casprep := '6' end
                                end;
                                if (((lignefamml.catgram='1')and((lignefamml.cas_pers_nbr=cass)
                                                                 or(lignefamml.cas_pers_nbr=casp)))
                                 or ((lignefamml.catgram='2')and((lignefamml.cas_pers_nbr=cass)
                                                                 or(lignefamml.cas_pers_nbr=casp)))
                                 or ((lignefamml.catgram='4')and((lignefamml.cas_pers_nbr=cass)
                                                                 or(lignefamml.cas_pers_nbr=casp)))
                                or ((lignefamml.catgram='8')and(lignefamml.sscatgram_dgr_vx='1'))
                               or ((lignefamml.catgram='7')and(lignefamml.cas_pers_nbr=casprep)))
                                 then begin
                                        trouve := true;
                                        new (ln3);
                                        ln3^.mot := lignefamml.motlat;
                                        ln3^.libre := true;
                                        ln3^.suivant := nil;
                                        if (ln3cour = nil)
                                        then begin
                                                firstln3 := ln3;
                                                ln3cour := ln3
                                             end
                                        else begin
                                                ln3cour^.suivant := ln3;
                                                ln3cour := ln3cour^.suivant
                                             end
                                    end
                                end
                        end
                    end;
                mcpcour := mcpcour^.suivant
            end
        end;
ln3cour := firstln3;
while (ln3cour <> nil) do
begin
     gotoxy (1, 12);
     delline;
     stop := false;
     k := 1;
     repeat
           if (ln3cour^.mot[k] = ' ')
           then stop := true
           else k := k + 1
     until ( (k > 20) or (stop = true) );
     longreelle := k - 1;
     motaecrire := copy (ln3cour^.mot, 1, longreelle);
     case casgn of
     nom : write ('Acceptez-vous l''{l{ment suivant du GNsujet ? ''O''-''N'' : ', motaecrire, ' : _');
     acc : write ('Acceptez-vous l''{l{ment suivant du GNcod ? ''O''-''N'' : ', motaecrire,' : _');
    abl : write ('Acceptez-vous l''{l{ment suivant du GNcompl{ment ? ''O''-''N'' : ',motaecrire,' : _')
     end;
     gotoxy (wherex - 1, wherey);
```

```
                    ok := false;
                    while (ok = false) do
                    begin
                         if keypressed
                         then begin
                                   ch := readkey;
                                   case ch of
                                        'O', 'o' : begin
                                                        write (ch);
                                                        ok := true;
                                                        ln3cour^.libre := true
                                                   end;
                                        'N', 'n' : begin
                                                        write (ch);
                                                        ok := true;
                                                        ln3cour^.libre := false
                                                   end;
                                        else       begin
                                                        sound (220);
                                                        delay (200);
                                                        nosound
                                                   end
                                   end
                              end
                    end;
                    ln3cour := ln3cour^.suivant
               end;
               AFFICHLN3 (firstln3, 1, 8)
          end
end;

if (noeudfilscour^.noeud.ind_brk = indicatif)
then begin
          (* MAJLIBRE pour tous les mots du GN *)
          for i := 1 to 85 do
          begin
               if (noeudfilscour^.noeud.ensmots3[i] = true)
               then begin
                         mcpcour := firstmcp;
                         k := 1;
                         while (k < i) do begin
                                             mcpcour := mcpcour^.suivant;
                                             k := k + 1
                                        end;
                         if (mcpcour^.libre = true)
                         then mcpcour^.libre := false
                         else begin
                                   mememot := mcpcour^.mot;
                                   repeat
                                        mcpcour := mcpcour^.suivant
                                   until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = true) );
                                   mcpcour^.libre := false
                              end
                    end
          end
     end
else begin
          MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
          if (remontee = true)
          then if (fin = true)
               then begin
                         finanbool := true;
                         finanal := true;
                         goto FINTRTGN
                    end
```

```
                else case casgn of
                        nom : if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.fonction = sujet) )
                              then goto ANAL
                              else if ((lnacour^.noeud.typenoeud = n2)and(lnacour^.noeudpere^.noeud.fonction = sujet)
                                   then goto CHOIXGN
                                   else begin
                                             trtsuivant := trtv;
                                             goto FINTRTGN
                                        end;
                        acc : if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.fonction = cod) )
                              then goto ANAL
                              else if ((lnacour^.noeud.typenoeud = n2)and(lnacour^.noeudpere^.noeud.fonction = cod))
                                   then goto CHOIXGN
                                   else begin
                                             trtsuivant := trtgns;
                                             goto FINTRTGN
                                        end;
                        abl : if ( (lnacour^.noeud.typenoeud = n1) and (lnacour^.noeud.fonction = cplt) )
                              then goto ANAL
                              else if ((lnacour^.noeud.typenoeud = n2)and(lnacour^.noeudpere^.noeud.fonction = cplt))
                                   then goto CHOIXGN
                                   else begin
                                             trtsuivant := trtgncod;
                                             goto FINTRTGN
                                        end
                    end
        end;
(* analyses morphologiques des (l(ments du GN *)
    TRSF_N3_LN3 (firstln3, noeudfilscour^.noeud.ensmots3, firstmcp);
    ln3cour := firstln3;
ANALGN :
    while (ln3cour <> nil) do
    begin
         ajout := false;
         if (gotoanalgn = true)
         then gotoanalgn := false
         else
              if (existfna = false)
              then begin
                        lnacour := lnacour^.noeudfils;
                        trouve := false;
                        while ( (lnacour <> nil) and (trouve = false) )
                        do if (lnacour^.noeud.ind_brk = indicatif)
                           then trouve := true
                           else lnacour := lnacour^.noeudfrere
                   end
              else begin
                        lnacour := lnacour^.noeudfils;
                        trouve := false;
                        stop := false;
                        while ( (lnacour <> nil) and (trouve = false) and (stop = false) )
                        do if (lnacour^.noeudfils <> nil)
                           then case casgn of
                                     nom :
                           if ((lnacour^.noeudfils^.noeud.typenoeud = n1) and (lnacour^.noeudfils^.noeud.fonction = sujet))
                           then trouve := true
                           else stop := true;
                                     acc :
                           if ((lnacour^.noeudfils^.noeud.typenoeud = n1) and (lnacour^.noeudfils^.noeud.fonction = cod))
                           then trouve := true
                           else stop := true;
                                     abl :
                           if ((lnacour^.noeudfils^.noeud.typenoeud = n1) and (lnacour^.noeudfils^.noeud.fonction = cplt))
                           then trouve := true
                           else stop := true
```

```
                        end
                    else if (lnacour^.noeud.ind_brk = indicatif)
                        then begin
                                trouve := true;
                                existfna := false
                            end
                        else lnacour := lnacour^.noeudfrere;
                    if (trouve = false) then existfna := false;
                    if (stop = true) then begin
                                            noeudmem := lnacour^.noeudfils;
                                            existfna := false
                                        end
                end;
DEBUT :
        if (existfna = true)
        then begin
                noeudfilscour := lnacour^.noeudfils;
                if (noeudfilscour^.noeud.mot1 = ln3cour^.mot)
                then begin
                        lnacour := lnacour^.noeudfils;
                        goto AFFICH
                    end
                else begin
                        ln3temp := ln3cour^.suivant;
                        trouve := false;
                        while ( (ln3temp <> nil) and (trouve = false) )
                        do if (ln3temp^.mot = noeudfilscour^.noeud.mot1)
                           then trouve := true
                           else ln3temp := ln3temp^.suivant;
                        if (trouve = false)
                        then begin
                                lnatemp := lnacour^.noeudfils;
                                noeudfilscour := lnatemp^.noeudfils;
                                while ((noeudfilscour^.noeudfils = nil)and(noeudfilscour^.noeud.ind_brk = break))
                                do noeudfilscour := noeudfilscour^.noeudfrere;
                                lnatemp := noeudfilscour^.noeudfils;
                                lnacour^.noeudfils := lnatemp;
                                goto DEBUT
                            end
                        else begin
                                new (newlna);
                                newlna^.noeudfils := noeudfilscour;
                                newlna^.noeudfrere := nil;
                                newlna^.noeudpere := lnacour;
                                newlna^.noeud.appart_analyse_correct := true;
                                newlna^.noeud.typenoeud := ni;
                                case casgn of
                                     nom : newlna^.noeud.fonction := sujet;
                                     acc : newlna^.noeud.fonction := cod;
                                     abl : newlna^.noeud.fonction := cplt
                                end;
                                newlna^.noeud.mot1 := ln3cour^.mot;
                     newlna^.noeud.comment_syst := concat ('choix {l{ment appartenant au GN pour analyse : ', ln3cour^.mot);
                                newlna^.noeud.comment_specif := '-';
                                newlna^.noeud.accept_comment := false;
                                newlna^.noeud.ind_brk := indicatif;
                                lnacour^.noeudfils := newlna;
                                lnacour := lnacour^.noeudfils;
                                ajout := true
                            end
                    end
            end
        else begin
                new (newlna);
                newlna^.noeudfils := nil;
```

```
            newlna^.noeudfrere := nil;
            newlna^.noeudpere := lnacour;
            newlna^.noeud.appart_analyse_correct := true;
            newlna^.noeud.typenoeud := n1;
            case casgn of
                nom : newlna^.noeud.fonction := sujet;
                acc : newlna^.noeud.fonction := cod;
                abl : newlna^.noeud.fonction := cplt
            end;
            newlna^.noeud.mot1 := ln3cour^.mot;
        newlna^.noeud.comment_syst := concat ('choix {l{ment appartenant au GN pour analyse : ', ln3cour^.mot);
            newlna^.noeud.comment_specif := '-';
            newlna^.noeud.accept_comment := false;
            newlna^.noeud.ind_brk := indicatif;
            lnacour^.noeudfils := newlna;
            lnacour := lnacour^.noeudfils;
        end;
    noeudfilscour := lnacour^.noeudfils;
    if (accesfamml = false) then begin
                                    assign (famml, 'b:fmotlat.am');
                                    reset (famml);
                                    accesfamml := true
                                end;
    FIND_INDEX (ln3cour^.mot, position);
    seek (famml, position);
    egalite := true;
    while ( (not eof(famml)) and (egalite = true) ) do
    begin
        read (famml, lignefamml);
        if (lignefamml.motlat <> ln3cour^.mot)
        then egalite := false
        else begin
            case casgn of
                nom : begin cass := 'A'; casp := 'J'; casprep := '1' end;
                acc : begin cass := 'C'; casp := 'L'; casprep := '3' end;
                abl : begin cass := 'F'; casp := 'O'; casprep := '6' end
            end;
            if ( ( (lignefamml.catgram = '1') and ((lignefamml.cas_pers_nbr = cass)
                                                or (lignefamml.cas_pers_nbr = casp)) )
              or ( (lignefamml.catgram = '2') and ((lignefamml.cas_pers_nbr = cass)
                                                or (lignefamml.cas_pers_nbr = casp)) )
              or ( (lignefamml.catgram = '4') and ((lignefamml.cas_pers_nbr = cass)
                                                or (lignefamml.cas_pers_nbr = casp)) )
              or ( (lignefamml.catgram = '8') and (lignefamml.sscatgram_dgr_vx = '1') )
              or ( (lignefamml.catgram = '7') and (lignefamml.cas_pers_nbr = casprep) ) )
            then begin
                    new (newlna);
                    newlna^.noeudfils := nil;
                    newlna^.noeudfrere := nil;
                    newlna^.noeudpere := lnacour;
                    newlna^.noeud.appart_analyse_correct := true;
                    newlna^.noeud.typenoeud := n2;
                    newlna^.noeud.lemme := lignefamml.lemme;
                    case lignefamml.catgram of
                        '1' : begin
                                  newlna^.noeud.categorie := subst;
                                  REMPLI_AM_SUBST (newlna^.noeud, lignefamml)
                              end;
                        '2' : begin
                                  newlna^.noeud.categorie := adj;
                                  REMPLI_AM_ADJ (newlna^.noeud, lignefamml)
                              end;
                        '4' : begin
                                  newlna^.noeud.categorie := adjpron;
                                  REMPLI_AM_ADJPRON (newlna^.noeud, lignefamml)
```

```
                                    end;
                               '7' : begin
                                          newlna^.noeud.categorie := prep;
                                          REMPLI_AM_PREP (newlna^.noeud, lignefamml)
                                    end;
                               '8' : begin
                                          newlna^.noeud.categorie := conj;
                                          REMPLI_AM_CONJ (newlna^.noeud, lignefamml)
                                    end
                          end;
                          newlna^.noeud.comment_syst := concat ('Lemme : ', lignefamml.lemme);
                          if (ajout = true)
                          then if (noeudfilscour^.noeud.typenoeud = nil)
                               then begin
                                          newlna^.noeudfils := lnacour^.noeudfils;
                                          lnacour^.noeudfils := newlna;
                                          noeudfilscour := lnacour^.noeudfils
                                    end
                               else begin
                                          noeudfilscour^.noeudfrere := newlna;
                                          noeudfilscour := noeudfilscour^.noeudfrere
                                    end
                          else if (noeudfilscour = nil)
                               then begin
                                          lnacour^.noeudfils := newlna;
                                          noeudfilscour := lnacour^.noeudfils
                                    end
                               else begin
                                          noeudfilscour^.noeudfrere := newlna;
                                          noeudfilscour := noeudfilscour^.noeudfrere
                                    end
                     end
                end
          end;
AFFICH :
          noeudfilscour := lnacour^.noeudfils;
          cheminok := false;
          while (noeudfilscour <> nil) do
          begin
               clrscr;
               gotoxy (1, 2);
               case casgn of
                    nom : write ('Analyse de l''{l{ment suivant appartenant au GNsujet : ', lnacour^.noeud.mot1);
                    acc : write ('Analyse de l''{l{ment suivant appartenant au GNcod : ', lnacour^.noeud.mot1);
                    abl : write ('Analyse de l''{l{ment suivant appartenant au GNcompl{ment : ', lnacour^.noeud.mot1)
               end;
               case noeudfilscour^.noeud.categorie of
                    subst : AFFICH_AM_SUBST (4, noeudfilscour^.noeud);
                    adj   : AFFICH_AM_ADJ (4, noeudfilscour^.noeud);
                    adjpron : AFFICH_AM_ADJPRON (4, noeudfilscour^.noeud);
                    conj  : AFFICH_AM_CONJ (4, noeudfilscour^.noeud);
                    prep : AFFICH_AM_PREP (4, noeudfilscour^.noeud)
               end;
               gotoxy (1, 9);
               writeln ('Commentaire du syst}me :');
               write (noeudfilscour^.noeud.comment_syst);
               AFFCOMM (11, noeudfilscour^.noeud.comment_specif, existfna);
               AFFACCCOMM (13, noeudfilscour^.noeud.accept_comment, existfna);
               AFFINDBRK (15, noeudfilscour, existfna, cheminok);
               gotoxy (40, 15);
               write ('Tapez une touche pour continuer ou ESC');
               repeat until keypressed;
               ch := readkey;
               if (ch = #27) then begin
                                       finanbool := true;
```

```
                                goto FINTRTGN
                            end;
                noeudfilscour := noeudfilscour^.noeudfrere
            end;
            if (ajout = true)
            then begin
                    noeudfilscour := lnacour^.noeudfils;
                    noeudsuivant := noeudfilscour^.noeudfils;
                    while (noeudfilscour^.noeud.ind_brk = break)
                    do noeudfilscour := noeudfilscour^.noeudfrere;
                    noeudfilscour^.noeudfils := noeudsuivant;
                    if (lnacour^.noeudfils^.noeud.ind_brk = break)
                    then lnacour^.noeudfils^.noeudfils := nil
                end;
            ln3cour := ln3cour^.suivant
        end;
        MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fin);
        if (remontee = true)
        then if (fin = true)
            then begin
                    finanbool := true;
                    finanal := true;
                    goto FINTRTGN
                end
            else case lnacour^.noeud.typenoeud of
                    n1 : case casgn of
                            nom : if (lnacour^.noeud.fonction = sujet)
                                  then goto ANAL
                                  else begin
                                            trtsuivant := trtv;
                                            goto FINTRTGN
                                       end;
                            acc : if (lnacour^.noeud.fonction = cod)
                                  then goto ANAL
                                  else begin
                                            trtsuivant := trtgns;
                                            goto FINTRTGN
                                       end;
                            abl : if (lnacour^.noeud.fonction = cplt)
                                  then goto ANAL
                                  else begin
                                            trtsuivant := trtgncod;
                                            goto FINTRTGN
                                       end
                        end;
                    n2 : case casgn of
                            nom : if ( ((lnacour^.noeudpere^.noeudpere^.noeud.typenoeud = n3)
                                    and (lnacour^.noeudpere^.noeudpere^.noeud.symbcatsynt = gnsujet))
                                    or (lnacour^.noeudpere^.noeudpere^.noeudpere^.noeud.fonction = sujet) )
                                  then begin
                                            lnatemp := lnacour;
                                            while (lnatemp^.noeud.typenoeud <> n3)
                                            do lnatemp := lnatemp^.noeudpere;
                                            TRSF_N3_LN3 (firstln3, lnatemp^.noeud.ensmots3, firstmcp);
                                            ln3cour := firstln3;
                                            while (ln3cour^.mot <> lnacour^.noeudpere^.noeud.mot1)
                                            do ln3cour := ln3cour^.suivant;
                                            ln3cour := ln3cour^.suivant;
                                            if (ln3cour <> nil) then begin goto ANALGN; gotoanalgn := true end
                                                                else goto FINTRTGN
                                       end
                                  else if (lnacour^.noeudpere^.noeud.fonction <> sujet)
                                       then begin
                                                 trtsuivant := trtv;
                                                 goto FINTRTGN
```

```
                                end
                            else goto CHOIXGN;
                        acc : if ( ((lnacour^.noeudpere^.noeudpere^.noeud.typenoeud = n3)
                                and (lnacour^.noeudpere^.noeudpere^.noeud.symbcatsynt = gncod))
                                or (lnacour^.noeudpere^.noeudpere^.noeudpere^.noeud.fonction = cod) )
                              then begin
                                        lnatemp := lnacour;
                                        while (lnatemp^.noeud.typenoeud <> n3)
                                        do lnatemp := lnatemp^.noeudpere;
                                        TRSF_N3_LN3 (firstln3, lnatemp^.noeud.ensmots3, firstmcp);
                                        ln3cour := firstln3;
                                        while (ln3cour^.mot <> lnacour^.noeudpere^.noeud.mot1)
                                        do ln3cour := ln3cour^.suivant;
                                        ln3cour := ln3cour^.suivant;
                                        if (ln3cour <> nil) then begin goto ANALGN; gotoanalgn := true end
                                                            else goto FINTRTGN
                                    end
                                else if (lnacour^.noeudpere^.noeud.fonction <> cod)
                                    then begin
                                              trtsuivant := trtgns;
                                              goto FINTRTGN
                                         end
                                    else goto CHOIXGN;
                        abl : if ( ((lnacour^.noeudpere^.noeudpere^.noeud.typenoeud = n3)
                                and (lnacour^.noeudpere^.noeudpere^.noeud.symbcatsynt = gnabl))
                                or (lnacour^.noeudpere^.noeudpere^.noeudpere^.noeud.fonction = cplt) )
                              then begin
                                        lnatemp := lnacour;
                                        while (lnatemp^.noeud.typenoeud <> n3)
                                        do lnatemp := lnatemp^.noeudpere;
                                        TRSF_N3_LN3 (firstln3, lnatemp^.noeud.ensmots3, firstmcp);
                                        ln3cour := firstln3;
                                        while (ln3cour^.mot <> lnacour^.noeudpere^.noeud.mot1)
                                        do ln3cour := ln3cour^.suivant;
                                        ln3cour := ln3cour^.suivant;
                                        if (ln3cour <> nil) then begin goto ANALGN; gotoanalgn := true end
                                                            else goto FINTRTGN
                                    end
                                else if (lnacour^.noeudpere^.noeud.fonction <> cplt)
                                    then begin
                                              trtsuivant := trtgncod;
                                              goto FINTRTGN
                                         end
                                    else goto CHOIXGN
                            end
                    end;
FINTRTGN :
    write ('')
end;

End.
```

```
Unit TRTTRAD;

Interface
uses ('$U b:globald2 ') GLOBALDECL2, CRT, DOS, ('$U b:lp_unit ') LP_UNIT,
     ('$U b:cpltproc ') CPLTPROC;

type str10 = string[10];

procedure TRSF (var noeud : noeudaa; var analyse : str10);

procedure TRT_TRAD (existfna : boolean; var firstmcp : lcontphr; var accesfamml : boolean;
                    var lnacour : listnoeudaa; var finanbool : boolean);

Implementation

procedure TRSF;

begin
      analyse := '          ';
      case noeud.categorie of
           subst : begin
                        analyse[1] := '1';
                        case noeud.declsubst of
                             d1 : analyse[2] := '1';
                             d2 : analyse[2] := '2';
                             d3 : analyse[2] := '3';
                             d4 : analyse[2] := '4';
                             d5 : analyse[2] := '5';
                             d6 : analyse[2] := '6';
                             d7 : analyse[2] := '7';
                        end;
                        case noeud.cassubst of
                             nom : case noeud.nbrsubst of
                                        sing : analyse[3] := 'A';
                                        pluriel : analyse[3] := 'J'
                                   end;
                             voc : case noeud.nbrsubst of
                                        sing : analyse[3] := 'B';
                                        pluriel : analyse[3] := 'K'
                                   end;
                             acc : case noeud.nbrsubst of
                                        sing : analyse[3] := 'C';
                                        pluriel : analyse[3] := 'L'
                                   end;
                             gen : case noeud.nbrsubst of
                                        sing : analyse[3] := 'D';
                                        pluriel : analyse[3] := 'M'
                                   end;
                             datif : case noeud.nbrsubst of
                                          sing : analyse[3] := 'E';
                                          pluriel : analyse[3] := 'N'
                                     end;
                             abl : case noeud.nbrsubst of
                                        sing : analyse[3] := 'F';
                                        pluriel : analyse[3] := 'O'
                                   end;
                             loc : case noeud.nbrsubst of
                                        sing : analyse[3] := 'G';
                                        pluriel : analyse[3] := 'P'
                                   end;
                             indecl : analyse[3] := 'Z'
                        end;
                        analyse[4] := '0';
                        analyse[5] := '0';
                        analyse[6] := ' ';
```

```
                    analyse[7] := ' ';
                    case noeud.genresubst of
                        commun : analyse[8] := '1';
                        fem : analyse[8] := '2';
                        masc fem : analyse[8] := '3';
                        masc : analyse[8] := '4';
                        mascneutre : analyse[8] := '5';
                        neutre : analyse[8] := '6';
                        genrenul : analyse[8] := ' ';
                    end;
                    analyse[9] := ' ';
                    analyse[10] := ' ';
            end;
        adj : begin
                    analyse[1] := '2';
                    case noeud.classeadj of
                        a1 : case noeud.degreadj of
                                positif : analyse[2] := '1';
                                comparatif : analyse[2] := 'A';
                                superlatif : analyse[2] := 'J'
                             end;
                        a2 : case noeud.degreadj of
                                positif : analyse[2] := '2';
                                comparatif : analyse[2] := 'B';
                                superlatif : analyse[2] := 'K'
                             end;
                        a3 : case noeud.degreadj of
                                positif : analyse[2] := '3';
                                comparatif : analyse[2] := 'C';
                                superlatif : analyse[2] := 'L'
                             end;
                        a4 : case noeud.degreadj of
                                positif : analyse[2] := '4';
                                comparatif : analyse[2] := 'D';
                                superlatif : analyse[2] := 'M'
                             end;
                        a5 : case noeud.degreadj of
                                positif : analyse[2] := '5';
                                comparatif : analyse[2] := 'E';
                                superlatif : analyse[2] := 'N'
                             end;
                        a6 : case noeud.degreadj of
                                positif : analyse[2] := '6';
                                comparatif : analyse[2] := 'F';
                                superlatif : analyse[2] := 'O'
                             end;
                        a7 : analyse[2] := '7';
                    end;
                    case noeud.casadj of
                        nom : case noeud.nbradj of
                                sing : analyse[3] := 'A';
                                pluriel : analyse[3] := 'J'
                              end;
                        voc : case noeud.nbradj of
                                sing : analyse[3] := 'B';
                                pluriel : analyse[3] := 'K'
                              end;
                        acc : case noeud.nbradj of
                                sing : analyse[3] := 'C';
                                pluriel : analyse[3] := 'L'
                              end;
                        gen : case noeud.nbradj of
                                sing : analyse[3] := 'D';
                                pluriel : analyse[3] := 'M'
                              end;
```

```
                    datif : case noeud.nbradj of
                                sing : analyse[3] := 'E';
                                pluriel : analyse[3] := 'N'
                            end;
                    abl : case noeud.nbradj of
                              sing : analyse[3] := 'F';
                              pluriel : analyse[3] := 'O'
                          end;
                    loc : case noeud.nbradj of
                              sing : analyse[3] := 'G';
                              pluriel : analyse[3] := 'P'
                          end;
                    indecl : analyse[3] := 'Z'
                end;
                analyse[4] := '0';
                analyse[5] := '0';
                analyse[6] := ' ';
                analyse[7] := ' ';
                case noeud.genreadj of
                    commun : analyse[8] := '1';
                    fem : analyse[8] := '2';
                    mascfem : analyse[8] := '3';
                    masc : analyse[8] := '4';
                    mascneutre : analyse[8] := '5';
                    neutre : analyse[8] := '6';
                    genrenul : analyse[8] := ' ';
                end;
                analyse[9] := ' ';
                analyse[10] := ' ';
        end;
num : begin
                analyse[1] := '3';
                case noeud.catnum of
                    card : analyse[2] := '1';
                    ord : case noeud.degrenum of
                              positif : analyse[2] := '2';
                              comparatif : analyse[2] := 'B';
                              superlatif : analyse[2] := 'K'
                          end;
                    distr : analyse[2] := '3';
                    mult : analyse[2] := '4';
                    advord : case noeud.degrenum of
                                 positif : analyse[2] := '5';
                                 comparatif : analyse[2] := 'E';
                                 superlatif : analyse[2] := 'N'
                             end;
                    advmult : analyse[2] := '6'
                end;
                case noeud.casnum of
                    nom : case noeud.nbrnum of
                              sing : analyse[3] := 'A';
                              pluriel : analyse[3] := 'J'
                          end;
                    voc : case noeud.nbrnum of
                              sing : analyse[3] := 'B';
                              pluriel : analyse[3] := 'K'
                          end;
                    acc : case noeud.nbrnum of
                              sing : analyse[3] := 'C';
                              pluriel : analyse[3] := 'L'
                          end;
                    gen : case noeud.nbrnum of
                              sing : analyse[3] := 'D';
                              pluriel : analyse[3] := 'M'
                          end;
```

```
                    datif : case noeud.nbrnum of
                                sing : analyse[3] := 'E';
                                pluriel : analyse[3] := 'N'
                            end;
                    abl : case noeud.nbrnum of
                                sing : analyse[3] := 'F';
                                pluriel : analyse[3] := 'O'
                          end;
                    loc : case noeud.nbrnum of
                                sing : analyse[3] := 'G';
                                pluriel : analyse[3] := 'P'
                          end;
                    indecl : analyse[3] := 'Z'
                end;
                analyse[4] := '0';
                analyse[5] := '0';
                analyse[6] := ' ';
                analyse[7] := ' ';
                case noeud.genrenum of
                    commun : analyse[8] := '1';
                    fem : analyse[8] := '2';
                    mascfem : analyse[8] := '3';
                    masc : analyse[8] := '4';
                    mascneutre : analyse[8] := '5';
                    neutre : analyse[8] := '6';
                    genrenul : analyse[8] := ' ';
                end;
                analyse[9] := ' ';
                analyse[10] := ' ';
        end;
adjpron : begin
                analyse[1] := '4';
                case noeud.catadjpron of
                    pers : analyse[2] := '1';
                    pos : analyse[2] := '2';
                    refl : analyse[2] := '3';
                    posrefl : analyse[2] := '4';
                    dem : analyse[2] := '5';
                    relat : analyse[2] := '6';
                    inter : analyse[2] := '7';
                    ind : analyse[2] := '8'
                end;
                case noeud.casadjpron of
                    nom : case noeud.nbradjpron of
                                sing : analyse[3] := 'A';
                                pluriel : analyse[3] := 'J'
                          end;
                    voc : case noeud.nbradjpron of
                                sing : analyse[3] := 'B';
                                pluriel : analyse[3] := 'K'
                          end;
                    acc : case noeud.nbradjpron of
                                sing : analyse[3] := 'C';
                                pluriel : analyse[3] := 'L'
                          end;
                    gen : case noeud.nbradjpron of
                                sing : analyse[3] := 'D';
                                pluriel : analyse[3] := 'M'
                          end;
                    datif : case noeud.nbradjpron of
                                sing : analyse[3] := 'E';
                                pluriel : analyse[3] := 'N'
                            end;
                    abl : case noeud.nbradjpron of
                                sing : analyse[3] := 'F';
```

```
                    pluriel : analyse[3] := 'O'
                end;
            loc : case noeud.nbradjpron of
                    sing : analyse[3] := 'G';
                    pluriel : analyse[3] := 'P'
                end;
            indecl : analyse[3] := 'Z'
        end;
        case noeud.modesubadjpron of
            indic : analyse[4] := '1';
            imp : analyse[4] := '2';
            subj : analyse[4] := '3';
            part : analyse[4] := '4';
            adjverb : analyse[4] := '5';
            gerondif : analyse[4] := '6';
            inf : analyse[4] := '7';
            supinum : analyse[4] := '8';
            supinu : analyse[4] := '9';
            modenul : if ( (analyse[2] = '6') or (analyse[2] = '7') )
                      then analyse[4] := ' '
                      else analyse[4] := '0'
        end;
        case noeud.tempssubadjpron of
            pres : analyse[5] := '1';
            impft : analyse[5] := '2';
            futspl : analyse[5] := '3';
            prft : analyse[5] := '4';
            plusqueprft : analyse[5] := '5';
            futant : analyse[5] := '6';
            fuisse : analyse[5] := '7';
            tempsnul : if ( (analyse[2] = '6') or (analyse[2] = '7') )
                       then analyse[5] := ' '
                       else analyse[5] := '0'
        end;
        analyse[6] := ' ';
        analyse[7] := ' ';
        case noeud.genreadjpron of
            commun : analyse[8] := '1';
            fem : analyse[8] := '2';
            mascfem : analyse[8] := '3';
            masc : analyse[8] := '4';
            mascneutre : analyse[8] := '5';
            neutre : analyse[8] := '6';
            genrenul : analyse[8] := ' ';
        end;
        analyse[9] := ' ';
        analyse[10] := ' ';
    end;
v : begin
    if (noeud.lemme = 'SVM             ')
    then analyse[1] := 'E'
    else analyse[1] := '5';
    case noeud.conjv of
        c1 : case noeud.voixv of
                actif : analyse[2] := '1';
                passif : analyse[2] := 'A';
                deponent : analyse[2] := 'J'
             end;
        c2 : case noeud.voixv of
                actif : analyse[2] := '2';
                passif : analyse[2] := 'B';
                deponent : analyse[2] := 'K';
                semideponent : analyse[2] := 'S'
             end;
        c3 : case noeud.voixv of
```

```
                    actif : analyse[2] := '3';
                    passif : analyse[2] := 'C';
                    deponent : analyse[2] := 'L';
                    semideponent : analyse[2] := 'T'
               end;
          c4 : case noeud.voixv of
                    actif : analyse[2] := '4';
                    passif : analyse[2] := 'D';
                    deponent : analyse[2] := 'M'
               end;
          c5 : case noeud.voixv of
                    actif : analyse[2] := '5';
                    passif : analyse[2] := 'E';
                    deponent : analyse[2] := 'N'
               end;
          c6 : case noeud.voixv of
                    actif : analyse[2] := '6';
                    passif : analyse[2] := 'F';
                    deponent : analyse[2] := 'O'
               end
     end;
     case noeud.modev of
          indic : begin
                       analyse[4] := '1';
                       case noeud.persv of
                            p1 : case noeud.nbrlv of
                                      sing : analyse[3] := 'A';
                                      pluriel : analyse[3] := 'J'
                                 end;
                            p2 : case noeud.nbrlv of
                                      sing : analyse[3] := 'B';
                                      pluriel : analyse[3] := 'K'
                                 end;
                            p3 : case noeud.nbrlv of
                                      sing : analyse[3] := 'C';
                                      pluriel : analyse[3] := 'L'
                                 end
                       end
                  end;
          imp : begin
                       analyse[4] := '2';
                       case noeud.persv of
                            p1 : case noeud.nbrlv of
                                      sing : analyse[3] := 'A';
                                      pluriel : analyse[3] := 'J'
                                 end;
                            p2 : case noeud.nbrlv of
                                      sing : analyse[3] := 'B';
                                      pluriel : analyse[3] := 'K'
                                 end;
                            p3 : case noeud.nbrlv of
                                      sing : analyse[3] := 'C';
                                      pluriel : analyse[3] := 'L'
                                 end
                       end
                  end;
          subj : begin
                       analyse[4] := '3';
                       case noeud.persv of
                            p1 : case noeud.nbrlv of
                                      sing : analyse[3] := 'A';
                                      pluriel : analyse[3] := 'J'
                                 end;
                            p2 : case noeud.nbrlv of
                                      sing : analyse[3] := 'B';
```

```
                                pluriel : analyse[3] := 'K'
                            end;
                        p3 : case noeud.nbr1v of
                                sing : analyse[3] := 'C';
                                pluriel : analyse[3] := 'L'
                            end
                    end
            end;
part : begin
                analyse[4] := '4';
                case noeud.casv of
                    nom : case noeud.nbr2v of
                                sing : analyse[3] := 'A';
                                pluriel : analyse[3] := 'J'
                            end;
                    voc : case noeud.nbr2v of
                                sing : analyse[3] := 'B';
                                pluriel : analyse[3] := 'K'
                            end;
                    acc : case noeud.nbr2v of
                                sing : analyse[3] := 'C';
                                pluriel : analyse[3] := 'L'
                            end;
                    gen : case noeud.nbr2v of
                                sing : analyse[3] := 'D';
                                pluriel : analyse[3] := 'M'
                            end;
                  datif : case noeud.nbr2v of
                                sing : analyse[3] := 'E';
                                pluriel : analyse[3] := 'N'
                            end;
                    abl : case noeud.nbr2v of
                                sing : analyse[3] := 'F';
                                pluriel : analyse[3] := 'O'
                            end;
                    loc : case noeud.nbr2v of
                                sing : analyse[3] := 'G';
                                pluriel : analyse[3] := 'P'
                            end;
                  indecl : analyse[3] := 'Z'
                end
        end;
adjverb : begin
                analyse[4] := '5';
                case noeud.casv of
                    nom : case noeud.nbr2v of
                                sing : analyse[3] := 'A';
                                pluriel : analyse[3] := 'J'
                            end;
                    voc : case noeud.nbr2v of
                                sing : analyse[3] := 'B';
                                pluriel : analyse[3] := 'K'
                            end;
                    acc : case noeud.nbr2v of
                                sing : analyse[3] := 'C';
                                pluriel : analyse[3] := 'L'
                            end;
                    gen : case noeud.nbr2v of
                                sing : analyse[3] := 'D';
                                pluriel : analyse[3] := 'M'
                            end;
                  datif : case noeud.nbr2v of
                                sing : analyse[3] := 'E';
                                pluriel : analyse[3] := 'N'
                            end;
```

```
                        abl : case noeud.nbr2v of
                                  sing : analyse[3] := 'F';
                                  pluriel : analyse[3] := 'O'
                              end;
                        loc : case noeud.nbr2v of
                                  sing : analyse[3] := 'G';
                                  pluriel : analyse[3] := 'P'
                              end;
                     indecl : analyse[3] := 'Z'
                    end
              end;
       gerondif : begin
                    analyse[4] := '6';
                    case noeud.casv of
                        nom : case noeud.nbr2v of
                                  sing : analyse[3] := 'A';
                                  pluriel : analyse[3] := 'J'
                              end;
                        voc : case noeud.nbr2v of
                                  sing : analyse[3] := 'B';
                                  pluriel : analyse[3] := 'K'
                              end;
                        acc : case noeud.nbr2v of
                                  sing : analyse[3] := 'C';
                                  pluriel : analyse[3] := 'L'
                              end;
                        gen : case noeud.nbr2v of
                                  sing : analyse[3] := 'D';
                                  pluriel : analyse[3] := 'M'
                              end;
                      datif : case noeud.nbr2v of
                                  sing : analyse[3] := 'E';
                                  pluriel : analyse[3] := 'N'
                              end;
                        abl : case noeud.nbr2v of
                                  sing : analyse[3] := 'F';
                                  pluriel : analyse[3] := 'O'
                              end;
                        loc : case noeud.nbr2v of
                                  sing : analyse[3] := 'G';
                                  pluriel : analyse[3] := 'P'
                              end;
                     indecl : analyse[3] := 'Z'
                    end
              end;
       inf : begin
                    analyse[4] := '7';
                    analyse[3] := '0'
             end;
       supinum : begin
                    analyse[4] := '8';
                    analyse[3] := '0'
                 end;
       supinu : begin
                    analyse[4] := '9';
                    analyse[3] := '0'
                end
end;
case noeud.tempsv of
     pres : analyse[5] := '1';
     impft : analyse[5] := '2';
     futspl : analyse[5] := '3';
     prft : analyse[5] := '4';
     plusqueprft : analyse[5] := '5';
     futant : analyse[5] := '6';
```

```
                fuisse : analyse[5] := '7';
                tempsnul : analyse[5] := '0'
            end;
            analyse[6] := ' ';
            analyse[7] := ' ';
            case noeud.genrev of
                commun : analyse[8] := '1';
                fem : analyse[8] := '2';
                mascfem : analyse[8] := '3';
                masc : analyse[8] := '4';
                mascneutre : analyse[8] := '5';
                neutre : analyse[8] := '6';
                genrenul : analyse[8] := ' ';
            end;
            analyse[9] := ' ';
            analyse[10] := ' '
        end;
    adv : begin
            analyse[1] := '6';
            case noeud.catadv of
                rel : analyse[2] := '6';
                int : analyse[2] := '7';
                neg : analyse[2] := '8';
                intneg : analyse[2] := '9';
                comp : analyse[2] := ' ';       (* ? *)
                superl : analyse[2] := '-';
                catnul : analyse[2] := '0'
            end;
            analyse[3] := '0';
            case noeud.modesubadv of
                indic : analyse[4] := '1';
                imp : analyse[4] := '2';
                subj : analyse[4] := '3';
                part : analyse[4] := '4';
                adjverb : analyse[4] := '5';
                gerondif : analyse[4] := '6';
                inf : analyse[4] := '7';
                supinum : analyse[4] := '8';
                supinu : analyse[4] := '9';
                modenul : if ( (analyse[2] = '6') or (analyse[2] = '7') )
                            then analyse[4] := ' '
                            else analyse[4] := '0'
            end;
            case noeud.tempssubadv of
                pres : analyse[5] := '1';
                impft : analyse[5] := '2';
                futspl : analyse[5] := '3';
                prft : analyse[5] := '4';
                plusqueprft : analyse[5] := '5';
                futant : analyse[5] := '6';
                fuisse : analyse[5] := '7';
                tempsnul : if ( (analyse[2] = '6') or (analyse[2] = '7') )
                             then analyse[5] := ' '
                             else analyse[5] := '0'
            end;
            analyse[6] := ' ';
            analyse[7] := ' ';
            analyse[8] := ' ';
            analyse[9] := ' ';
            analyse[10] := ' ';
        end;
    prep : begin
            analyse[1] := '7';
            case noeud.typemecum of
                true : analyse[2] := '1';
```

```
                    false : analyse[2] := '0'
                end;
                case noeud.casprep of
                    accusatif : analyse[3] := '3';
                    genitif : analyse[3] := '4';
                    ablatif : analyse[3] := '6'
                end;
                analyse[4] := '0';
                analyse[5] := '0';
                analyse[6] := ' ';
                analyse[7] := ' ';
                analyse[8] := ' ';
                analyse[9] := ' ';
                analyse[10] := ' ';
        end;
    conj : begin
                analyse[1] := '8';
                case noeud.catconj of
                    coord : analyse[2] := '1';
                    sub : analyse[2] := '2'
                end;
                analyse[3] := '0';
                case noeud.modesubconj of
                    indic : analyse[4] := '1';
                    imp : analyse[4] := '2';
                    subj : analyse[4] := '3';
                    part : analyse[4] := '4';
                    adjverb : analyse[4] := '5';
                    gerondif : analyse[4] := '6';
                    inf : analyse[4] := '7';
                    supinum : analyse[4] := '8';
                    supinu : analyse[4] := '9';
                    modenul : if (analyse[2] = '2')
                              then analyse[4] := ' '
                              else analyse[4] := '0'
                end;
                case noeud.tempssubconj of
                    pres : analyse[5] := '1';
                    impft : analyse[5] := '2';
                    futspl : analyse[5] := '3';
                    prft : analyse[5] := '4';
                    plusqueprft : analyse[5] := '5';
                    futant : analyse[5] := '6';
                    fuisse : analyse[5] := '7';
                    tempsnul : if (analyse[2] = '2')
                               then analyse[5] := ' '
                               else analyse[5] := '0'
                end;
                analyse[6] := ' ';
                analyse[7] := ' ';
                analyse[8] := ' ';
                analyse[9] := ' ';
                analyse[10] := ' ';
        end;
    interj : begin
                analyse[1] := '9';
                analyse[2] := '0';
                analyse[3] := '0';
                analyse[4] := '0';
                analyse[5] := '0';
                analyse[6] := ' ';
                analyse[7] := ' ';
                analyse[8] := ' ';
                analyse[9] := ' ';
                analyse[10] := ' '
```

```
                end
        end
end;

(*-------------------------------------------------------------------------*)

procedure TRT_TRAD;

var
    lnatemp, newlna, noeudfilscour : listnoeudaa;
    firstltflcour, ltflcour, newltfl, firstltfl, newltflcour, ltflcourprec : listtradfranc;
    analmorphol : str10;
    ch : char;
    stop, ok, accept, trouve, cheminok : boolean;
    longreelle, k, position : integer;
    motaecrire : str24;
    mcpcour : lcontphr;

label FINTRTTRAD, AFFICH;

begin
      finanbool := false;
(* recherche des traductions des mots s{lectionn{s selon une certaine A.M. *)
(* et cr{ation d'une liste pour affichage.                                  *)
      if (accesfamml = false) then begin
                                      assign (famml, 'b:fmotlat.am');
                                      reset (famml);
                                      accesfamml := true
                                   end;
      ltflcour := nil;
      lnatemp := lnacour;
      while (lnatemp <> nil) do
      begin
           case lnatemp^.noeud.typenoeud of
           n2 : begin
                     TRSF (lnatemp^.noeud, analmorphol);
                     FIND_INDEX (lnatemp^.noeudpere^.noeud.mot1, position);
                     seek (famml, position);
                     trouve := false;
                     while (trouve = false) do
                     begin
                          read (famml, lignefamml);
                          if ( (lignefamml.catgram = analmorphol[1])
                           and (lignefamml.sscatgram_dgr_vx = analmorphol[2])
                           and (lignefamml.cas_pers_nbr = analmorphol[3])
                           and (lignefamml.mode = analmorphol[4])
                           and (lignefamml.temps = analmorphol[5])
                           and (lignefamml.fonction = analmorphol[6])
                           and (lignefamml.emplois = analmorphol[7])
                           and (lignefamml.genre = analmorphol[8])
                           and (lignefamml.codesubord[1] = analmorphol[9])
                           and (lignefamml.codesubord[2] = analmorphol[10]) )
                          then begin
                                    trouve := true;
                                    new (newltfl);
                                    newltfl^.motlatin := lnatemp^.noeudpere^.noeud.mot1;
                                    newltfl^.tradfranc := lignefamml.tradf;
                                    newltfl^.suivant_tf := nil;
                                    if (ltflcour = nil)
                                    then begin
                                              firstltflcour := newltfl;
                                              ltflcour := newltfl
                                         end
                                    else begin
                                              ltflcour^.suivant_tf := newltfl;
```

```
                                        ltflcour := ltflcour^.suivant_tf
                                    end
                                end
                        end;
                        lnatemp := lnatemp^.noeudpere^.noeudpere
                    end;
            n3 : begin
                        lnatemp := lnatemp^.noeudpere^.noeudpere^.noeudpere
                    end
            end
    end;
(* classement de la liste selon l'ordre des mots de la phrase *)
    mcpcour := firstmcp;
    firstltfl := nil;
    newltflcour := nil;
    while (mcpcour <> nil) do
    begin
        ltflcour := firstltflcour;
        ltflcourprec := nil;
        while (ltflcour^.motlatin <> mcpcour^.mot) do
        begin
            ltflcour := ltflcour^.suivant_tf;
            if (ltflcourprec = nil) then ltflcourprec := firstltflcour
                                    else ltflcourprec := ltflcourprec^.suivant_tf
        end;
        new (newltfl);
        newltfl^.motlatin := ltflcour^.motlatin;
        newltfl^.tradfranc := ltflcour^.tradfranc;
        newltfl^.suivant_tf := nil;
        if (newltflcour = nil)
        then begin
                firstltfl := newltfl;
                newltflcour := newltfl
            end
        else begin
                newltflcour^.suivant_tf := newltfl;
                newltflcour := newltflcour^.suivant_tf
            end;
        if (ltflcourprec = nil) then firstltflcour := ltflcour^.suivant_tf
                                else ltflcourprec^.suivant_tf := ltflcour^.suivant_tf;
        mcpcour := mcpcour^.suivant
    end;
    if (existfna = false)
    then begin
            new (newlna);
            newlna^.noeudfils := nil;
            newlna^.noeudfrere := nil;
            newlna^.noeudpere := lnacour;
            newlna^.noeud.appart_analyse_correct := true;
            newlna^.noeud.typenoeud := n4;
            newlna^.noeud.comment_syst := '-';
            lnacour^.noeudfils := newlna
        end;
AFFICH :
(* affichage traductions *)
    noeudfilscour := lnacour^.noeudfils;
    accept := false;
    while (accept = false) do
    begin
        clrscr;
        ltflcour := firstltfl;
        while (ltflcour <> nil) do
        begin
            if (wherey > 8) then begin
                                    gotoxy (35, 10);
```

```
                                write ('Tapez une touche pour voir la page suivante ! ');
                                repeat until keypressed;
                                ch := readkey;
                                clrscr
                           end;
      stop := false;
      k := 24;
      while ( (stop = false) and (k > 0) ) do
      begin
           if (ltflcour^.tradfranc[k] <> ' ') then stop := true
                                              else k := k - 1
      end;
      longreelle := k;
      motaecrire := copy (ltflcour^.tradfranc, 1, longreelle);
      if ((80 - wherex) < longreelle) then writeln;
      write (motaecrire);
      if (wherex = 1) then gotoxy (80, wherey - 1)
                      else begin
                                write (' ');
                                if (wherex = 1) then gotoxy (80, wherey - 1)
                           end;
      ltflcour := ltflcour^.suivant_tf
end;
gotoxy (1, 10);
write ('Explications de cette ''traduction'' ? ''O''-''N'' : _');
gotoxy (wherex - 1, wherey);
ok := false;
while (ok = false) do
begin
      if keypressed then begin
                              ch := readkey;
                              case ch of
                                   'O', 'o', 'N', 'n' : begin
                                                             write (ch);
                                                             ok := true
                                                        end;
                                   else                 begin
                                                             sound (220);
                                                             delay (200);
                                                             nosound
                                                        end
                              end
                         end
end;
if ( (ch = 'N') or (ch = 'n') )
then begin
           accept := true;
           cheminok := false;
           gotoxy (1, 10);
           delline;
           writeln ('Commentaire du syst}me : ');
           write (noeudfilscour^.noeud.comment_syst);
           AFFCOMM (12, noeudfilscour^.noeud.comment_specif, existfna);
           AFFACCCOMM (14, noeudfilscour^.noeud.accept_comment, existfna);
           AFFINDBRK (15, noeudfilscour, existfna, cheminok);
           gotoxy (40, 15);
           write ('Tapez une touche pour continuer ou ESC');
           repeat until keypressed;
           ch := readkey;
           if (ch = #27) then begin
                                   finanbool := true;
                                   goto FINTRTTRAD
                              end
     end
else begin
```

```
                    ltflcour := firstltfl;
                    while (ltflcour <> nil) do
                    begin
                         gotoxy (1, 12);
                         delline;
                         stop := false;
                         k := 24;
                         while ( (stop = false) and (k > 0) ) do
                         begin
                              if (ltflcour^.tradfranc[k] <> ' ') then stop := true
                                                                 else k := k - 1
                         end;
                         longreelle := k;
                         motaecrire := copy (ltflcour^.tradfranc, 1, longreelle);
                         writeln ('Traduction de ', ltflcour^.motlatin, ' : ', motaecrire);
                         write ('Tapez une touche pour voir la suite');
                         repeat until keypressed;
                         ch := readkey;
                         ltflcour := ltflcour^.suivant_tf
                    end
               end
     end;
FINTRTTRAD :
     write ('')
end;

End.
```

```
Program LAT_PROF;

uses ('$U b:globald2 ')  GLOBALDECL2, CRT, DOS, ('$U b:lp_unit ') LP_UNIT,
     ('$U b:cpltproc ') CPLTPROC, ('$U b:trtvp ') TRTVP, ('$U b:trtgn ') TRTGN,
     ('$U b:trttrad ') TRTTRAD;

const
     colortxt = 14;       (* yellow *)
     colorfond = 0;      (* black *)

var
     out, gotoend, oke : boolean;
     ch                : char;
     namefct, namefdt  : str14;
     iocode            : integer;


label DEBUT;

(*************************************************************************)

procedure COMPLEMENTS (var namefct : str14);

(* sp{cifications : COMPLEMENTS permet de visualiser l'analyse existante des *)
(*                  phrases contenues dans le texte 'namefct' et de les     *)
(*                  compl{ter.                                              *)


var
    k, iocode, ymin, ymax, nbrephrase, i : integer;
    namefdt,nomfna : str14;
    titre : str60;
    finanbool, right, ok, oke, existfna, accesfamml, trouve, stop : boolean;
    remontev, remontegns, remontecod, remontecplt, remontee, fini, finanal : boolean;
    ch    : char;
    comment : str80;
    ldcour, firstld : list_descr;
    stri : string[3];
    firstmcp, mcpcour : lcontphr;
    firstlna, lnacour, noeudmem, noeudfilscour, lnatemp : listnoeudaa;
    trt, trtsuivant : traitement;

label FIN, FINAN, CHOIXPHR ;


Begin
     clrscr;
     accesfamml := false;
     k := length(namefct) - 4;
     namefdt := copy (namefct, 1, k);
     namefdt := concat (namefdt, '.dsc');
('$I-')
     assign (fdescrtxt, namefdt);
     reset (fdescrtxt);
     iocode := ioresult;
     if (iocode <> 0)
     then begin
               writeln (namefdt, ' n''existe pas ! Tapez une touche pour revenir au menu !');
               repeat until keypressed;
               ch := readkey;
               goto FIN
          end
     else begin
               nbrephrase := filesize (fdescrtxt) - 1;
               read (fdescrtxt, lignefdt);
```

```
                titre := lignefdt.titre_txt;
                comment := lignefdt.comment_txt;
                close (fdescrtxt);
                COPY_FDSC_LDSC (namefdt, firstld)
          end;
(*$I+*)
     ymin := 1;
     ymax := 19;
     VISUTXTPHR (namefct, 0, titre, ymin, ymax, right);

(* apr}s avoir vu le texte, il faudra afficher son commentaire, s'il existe *)
(* et proposer de le changer, laisser ou supprimer. Puis, on demandera de *)
(* s{lectionner la phrase @ analyser (possibilit{ de sortie ! ), ensuite on *)
(* g{rera aussi son commentaire et on passera alors aux {tapes d'analyse ! *)

     GESTCOMM (21, firstld, true);
CHOIXPHR :
     window (1, 1, 80, 25);
     clrscr;
     ok := false;
     i := 1;
     while (ok = false) do
     begin
          str (i, stri);
          titre := concat ('SELECTION DE LA PHRASE A ANALYSER : phrase n[', stri);
          VISUTXTPHR (namefct, i, titre, 5, 19, right);
          gotoxy (1, 21);
          write ('Voulez-vous analyser cette phrase ? ''O''ui, ''N''on ou ''ESC'' : ');
          oke := false;
          while (oke = false) do
          begin
               if keypressed
               then begin
                         ch := readkey;
                         case ch of
                              'O', 'o' : begin
                                              write (ch);
                                              oke := true;
                                              ok := true
                                         end;
                              'N', 'n' : begin
                                              write (ch);
                                              oke := true;
                                              i := i + 1;
                                              if (i > nbrephrase) then i := 1
                                         end;
                              #27      : goto FIN;
                              else       begin
                                              sound (220);
                                              delay (200);
                                              nosound
                                         end
                         end
                    end
          end
     end;
     delline;
     ldcour := firstld;
     k := 1;
     while (k <= i) do
     begin
          ldcour := ldcour^.suivant;
          k := k + 1
     end;
     GESTCOMM (21, ldcour, false);
```

```
    clrscr;
(* analyse de la phrase i *)
    if (ldcour^.contenu.etat_prep_phrase <> e3)
    then ldcour^.contenu.etat_prep_phrase := e2;
    ldcour := firstld;
    titre := ldcour^.contenu.titre_txt;
    str (i, stri);
    titre := concat (titre, ' phrase n[', stri);
    VISUTXTPHR (namefct, i, titre, 1, 10, right);
(* d{claration de la fene^tre d'analyse *)
    window (1, 11, 80, 25);
(* (tablissement d'une liste chai^n{e comprenant les mots de la phrase pour *)
(* pouvoir les manipuler plus facilement que via le fichier.              *)
    TRSF_PHR_L (namefct, i, firstmcp);

(* dans le cas o} l'analyse existe d{j@, il faut la reprendre !          *)
    reset (fdescrtxt);
    seek (fdescrtxt, i);
    read (fdescrtxt, lignefdt);
    existfna := false;
    if (lignefdt.nomfichanal <> '') then begin
                                          ($I-)
                                          nomfna := lignefdt.nomfichanal;
                                          assign (fna, lignefdt.nomfichanal);
                                          reset (fna);
                                          iocode := ioresult;
                                          if (iocode = 0)
                                          then begin
                                                    close (fna);
                                                    existfna := true;
                                                    COPY_FA_LA (lignefdt.nomfichanal, firstlna)
                                               end
                                          ($I+)
                                     end
                                else begin
                                          k := length (namefct) - 4;
                                          nomfna := copy (namefct, 1, k);
                                          nomfna := concat (nomfna, '.', stri);
                                          ldcour := firstld;
                                          k := 1;
                                          while (k <= i) do
                                          begin
                                               ldcour := ldcour^.suivant;
                                               k := k + 1
                                          end;
                                          ldcour^.contenu.nomfichanal := nomfna
                                     end;
    close (fdescrtxt);

    trt := trtv;
    stop := false;
    remontev := false;
    while (stop = false) do
    begin
         case trt of
              trtv : (* traitement du verbe principal *)
                     begin
                          TRT_VP (existfna, firstmcp, accesfamml, lnacour, firstlna, finanbool, remontev, finanal);
                          if (finanbool = true) then begin
                                                          stop := true;
                                                          if (finanal = true)
                                                          then begin
                                                                    (* maj de l'(tat de pr{paration : e3 *)
                                                                    ldcour := firstld;
                                                                    k := 1;
```

```
                                        while (k <= i) do
                                        begin
                                             ldcour := ldcour^.suivant;
                                             k := k + 1
                                        end;
                                        ldcour^.contenu.etat_prep_phrase := e3
                                   end
                              end
                         else begin
                                   trt := trtgns;
                                   remontegns := false
                              end

        end;
trtgns : (* traitement GNsujet : choix sujet, analyses morphologiques et d{coupe      *)
          (* syntaxique suivie de la s{lection et des analyses de chaque mot.        *)
          begin
            TRT_GN (existfna,firstmcp,accesfamml,lnacour,finanbool,noeudmem,nom,remontegns,trtsuivant,finanal);
            if (finanbool = true) then begin
                                        stop := true;
                                        if (finanal = true)
                                        then begin
                                                  (* maj de l'{tat de pr{paration : e3 *)
                                                  ldcour := firstld;
                                                  k := 1;
                                                  while (k <= i) do
                                                  begin
                                                       ldcour := ldcour^.suivant;
                                                       k := k + 1
                                                  end;
                                                  ldcour^.contenu.etat_prep_phrase := e3
                                             end
                                   end
                              else begin
                                        if (trtsuivant = trtnul)
                                        then begin
                                             (* si existfna = true then {liminer noeud n1/sujet + AM *)
                                               if (existfna = true)
                                               then begin
                                                     while ( (lnacour^.noeudfils <> nil) and
                                                          (lnacour^.noeudfils^.noeud.fonction = sujet) )
                                                     do begin
                                                          lnatemp := lnacour^.noeudfils;
                                                          noeudfilscour := lnatemp^.noeudfils;
                                                          while ( (noeudfilscour^.noeudfils = nil)
                                                              and (noeudfilscour^.noeud.ind_brk = break))
                                                          do noeudfilscour := noeudfilscour^.noeudfrere;
                                                          lnatemp := noeudfilscour^.noeudfils;
                                                          lnacour^.noeudfils := lnatemp
                                                        end;
                                                     if (lnacour^.noeudfils = nil) then existfna := false
                                                    end;
                                             (* si noeudmem <> nil alors le raccrocher *)
                                               if (noeudmem <> nil)
                                               then begin
                                                         lnacour^.noeudfils := noeudmem;
                                                         existfna := true
                                                    end;
                                               trt := trtgncod;
                                               remontecod := false
                                             end
                                        else begin
                                                  trt := trtsuivant;  (* trtv *)
                                                  remontev := true
                                             end
```

```
                                        end
                end;
trtgncod : (* traitement GNcod : choix cod, analyses morphologiques et d(coupe *)
            (* syntaxique suivie de la s(lection et des analyses de chaque mot. *)
            begin
            TRT_GN (existfna,firstmcp,accesfamml,lnacour,finanbool,noeudmem,acc,remontecod,trtsuivant,finanal);
            if (finanbool = true) then begin
                                        stop := true;
                                        if (finanal = true)
                                        then begin
                                                (* maj de l'(tat de pr(paration : e3 *)
                                                ldcour := firstld;
                                                k := 1;
                                                while (k <= i) do
                                                begin
                                                     ldcour := ldcour^.suivant;
                                                     k := k + 1
                                                end;
                                                ldcour^.contenu.etat_prep_phrase := e3
                                           end
                                    end
                                else begin
                                        if (trtsuivant = trtnul)
                                        then begin
                                             (* si existfna = true then (liminer noeud n1/cod + AM *)
                                               if (existfna = true)
                                               then begin
                                                      while ( (lnacour^.noeudfils <> nil) and
                                                           (lnacour^.noeudfils^.noeud.fonction = cod) )
                                                      do begin
                                                           lnatemp := lnacour^.noeudfils;
                                                           noeudfilscour := lnatemp^.noeudfils;
                                                           while ( (noeudfilscour^.noeudfils = nil)
                                                               and (noeudfilscour^.noeud.ind_brk = break))
                                                           do noeudfilscour := noeudfilscour^.noeudfrere;
                                                           lnatemp := noeudfilscour^.noeudfils;
                                                           lnacour^.noeudfils := lnatemp
                                                      end;
                                                      if (lnacour^.noeudfils = nil) then existfna := false
                                                end;
                                             (* si noeudmem <> nil alors le raccrocher *)
                                               if (noeudmem <> nil)
                                               then begin
                                                        lnacour^.noeudfils := noeudmem;
                                                        existfna := true
                                                end;
                                               trt := trtgncplt;
                                               remontecplt := false
                                             end
                                        else begin
                                                trt := trtsuivant;  (* trtgns *)
                                                remontegns := true
                                             end
                                    end
            end;
trtgncplt : (* traitement GNcompl(ment : choix cplt, analyses morphologiques et d(coupe *)
             (* syntaxique suivie de la s(lection et des analyses de chaque mot.        *)
             begin
            TRT_GN (existfna,firstmcp,accesfamml,lnacour,finanbool,noeudmem,abl,remontecplt,trtsuivant,finanal);
            if (finanbool = true) then begin
                                        stop := true;
                                        if (finanal = true)
                                        then begin
                                                (* maj de l'(tat de pr(paration : e3 *)
                                                ldcour := firstld;
```

```
                                        k := 1;
                                        while (k <= i) do
                                        begin
                                              ldcour := ldcour^.suivant;
                                              k := k + 1
                                        end;
                                        ldcour^.contenu.etat_prep_phrase := e3
                                  end
                            end
                      else begin
                                  if (trtsuivant = trtnul)
                                  then begin
                                       (* si existfna = true then {liminer noeud n1/cplt + AM *)
                                        if (existfna = true)
                                        then begin
                                              while ( (lnacour^.noeudfils <> nil) and
                                                   (lnacour^.noeudfils^.noeud.fonction = cplt) )
                                              do begin
                                                   lnatemp := lnacour^.noeudfils;
                                                   noeudfilscour := lnatemp^.noeudfils;
                                                   while ( (noeudfilscour^.noeudfils = nil)
                                                      and (noeudfilscour^.noeud.ind_brk = break))
                                                   do noeudfilscour := noeudfilscour^.noeudfrere;
                                                   lnatemp := noeudfilscour^.noeudfils;
                                                   lnacour^.noeudfils := lnatemp
                                                end;
                                              if (lnacour^.noeudfils = nil) then existfna := false
                                             end;
                                       (* si noeudmem <> nil alors le raccrocher *)
                                        if (noeudmem <> nil)
                                        then begin
                                                  lnacour^.noeudfils := noeudmem;
                                                  existfna := true
                                             end;
                                        trt := traduct
                                       end
                                  else begin
                                             trt := trtsuivant;  (* trtgncod *)
                                             remontecod := true
                                       end
                                end
                end;
traduct : (* traitement traductions *)
          begin
               TRT_TRAD (existfna, firstmcp, accesfamml, lnacour, finanbool);
               if (finanbool = true) then stop := true
                                     else begin
                                             MAJLNACOUR (lnacour, existfna, firstmcp, remontee, fini);
                                             if (remontee = true)
                                             then if (fini = true)
                                                  then begin
                                                            stop := true;
                                                            (* maj de l'{tat de pr{paration : e3 *)
                                                            ldcour := firstld;
                                                            k := 1;
                                                            while (k <= i) do
                                                            begin
                                                                  ldcour := ldcour^.suivant;
                                                                  k := k + 1
                                                            end;
                                                            ldcour^.contenu.etat_prep_phrase := e3
                                                       end
                                                  else begin
                                                            trt := trtgncplt;
                                                            remontecplt := true;
```

```
                                                        end
                                                    end
                        end
            end
    end;

(* recopie de la liste chai^n(e des noeuds d'analyse dans le fichier          *)
(* et retour @ l'endroit o¦ l'on peut choisir une autre phrase @ analyser     *)
FINAN:
    COPY_LA_FA (nomfna, firstlna);
    goto CHOIXPHR;

(* recopie de la liste chai^n(e de descriptions dans le fichier               *)
FIN :
    COPY_LDSC_FDSC (namefdt, firstld);
    if (accesfamml = true) then close (famml)
end;

(*****************************************************************************)

procedure TRT_AUT_COH;
begin
    clrscr;
    writeln ('procedure trt-aut-coh');
    repeat  until keypressed;
    ch := readkey
end;

(************************ d(but program prof_lat ******************************)

BEGIN
DEBUT :
    textcolor (colortxt);
    textbackground (colorfond);
    window (1,1,80,25);
    clrscr;
(* en ayant defini avant textbackground, clrscr permet de colorer l'ecran en cette couleur *)
(* sinon uniquement les parties ecrites de l'ecran sont colorees                           *)
    gotoxy (8,1);
    writeln ('DIDACTICIEL D''AIDE A L''ANALYSE ET LA TRADUCTION DE TEXTES LATINS :');
    gotoxy (8,2);
    writeln ('-------------------------------------------------------------');
    gotoxy (35,4);
    writeln( 'PREPARATION');
    gotoxy (35,5);
    writeln ('-----------');
    gotoxy (7,9);
    writeln('1- LISTE TEXTES LATINS');
    writeln;
    writeln ('      2- VISUALISATION TEXTE LATIN');
    writeln;
    writeln ('      3- COMPLEMENTS ANALYSE ET TRADUCTION TEXTE LATIN');
    writeln;
    writeln ('      4- TRAITEMENT AUTOMATIQUE : VERIFICATION COHERENCE ANALYSE TEXTE LATIN');
    writeln;
    writeln ('      5- LISTE RESULTATS TRAITEMENT AUTOMATIQUE');
    writeln;
    writeln ('      6- VISUALISATION ETAT PREPARATION TEXTE LATIN');
    writeln;
    writeln ('      7- FIN');
    gotoxy (18,25);
    write ('TAPEZ LE NUMERO CORRESPONDANT A VOTRE CHOIX : _');
    gotoxy (64,25);
    out := false;
    while ( out = false ) do
```

```
begin
     if keypressed
     then begin
               ch := readkey;
               case ch of
                    '1' : begin
                              write (ch);
                              LIST_TXT_LAT;
                              goto DEBUT
                         end;
                    '2' : begin
                              write (ch);
                              clrscr;
                              gotoend := false;
                              oke := false;
                              repeat
                                   write ('Nom du texte latin @ visualiser ou RETURN : b: _______.fct');
                                   gotoxy (48, wherey);
                                   readln (namefct);
                                   if namefct = ''
                                   then gotoend := true
                                   else if (VALID_TXT (namefct) = false)
                                        then writeln (namefct, ' n''existe pas !')
                                        else oke := true
                              until ( (oke = true) or (gotoend = true) );
                              if (gotoend <> true) then VISU_TXT_LAT(namefct);
                              goto DEBUT
                         end;
                    '3' : begin
                              write (ch);
                              clrscr;
                              gotoend := false;
                              oke := false;
                              repeat
                                   write ('Nom du texte latin dont l''analyse est @ compl{ter ou RETURN : ');
                                   write ('b: _______.fct');
                                   gotoxy (48, wherey);
                                   readln (namefct);
                                   if namefct = ''
                                   then gotoend := true
                                   else if (VALID_TXT (namefct) = false)
                                        then writeln (namefct, ' n''existe pas !')
                                        else oke := true
                              until ( (oke = true) or (gotoend = true) );
                              if (gotoend <> true) then COMPLEMENTS(namefct);
                              goto DEBUT
                         end;
                    '4' : begin
                              write (ch);
                              TRT_AUT_COH;
                              goto DEBUT
                         end;
                    '5' : begin
                              write (ch);
                              clrscr;
                              gotoend := false;
                              oke := false;
                              repeat
                                   writeln ('Nom du fichier contenant les r{sultats du traitement automatique ');
                                   write ('du texte latin associ{ ou RETURN : b: _______.dsc');
                                   gotoxy (39, wherey);
                                   readln (namefdt);
                                   if namefdt = ''
                                   then gotoend := true
                                   else begin
```

```
                                        namefdt := concat ('b:', namefdt, '.dsc');
(*$I-*)
                                        assign (fdescrtxt, namefdt);
                                        reset (fdescrtxt);
                                        iocode := ioresult;
                                        if (iocode <> 0 ) then writeln (namefdt, ' n''existe pas !')
                                    end
                            until ( (iocode = 0) or (gotoend = true) );
(*$I+*)
                            if (gotoend <> true) then begin
                                                        close (fdescrtxt);
                                                        LIST_RES_TRT_AUT(namefdt)
                                                    end;
                            goto DEBUT
                        end;
                    '6' : begin
                            write (ch);
                            clrscr;
                            gotoend := false;
                            oke := false;
                            repeat
                                writeln('Nom du fichier contenant les informations sur l''{tat de pr{paration');
                                write ('du texte latin associ{ ou RETURN : b: ________.dsc');
                                gotoxy (39, wherey);
                                readln (namefdt);
                                if namefdt = ''
                                then gotoend := true
                                else begin
                                        namefdt := concat ('b:', namefdt, '.dsc');
(*$I-*)
                                        assign (fdescrtxt, namefdt);
                                        reset (fdescrtxt);
                                        iocode := ioresult;
                                        if (iocode <> 0 ) then writeln (namefdt, ' n''existe pas !')
                                    end
                            until ( (iocode = 0) or (gotoend = true) );
(*$I+*)
                            if (gotoend <> true) then begin
                                                        close (fdescrtxt);
                                                        VISU_ETAT_PREP_TXT_LAT(namefdt)
                                                    end;
                            goto DEBUT
                        end;
                    '7' : begin
                            write (ch);
                            out := true
                        end;
                    else  begin
                            sound (220);
                            delay (200);
                            nosound
                        end
                end
            end
    end
END.
```

```
Unit GLOBALEL ;

Interface

   type
        traitement = (trtv, trtgns, trtgncod, trtgncplt, traduct, trtnul);
        str2 = string[2];
        str16 = string[16];
        str20 = string[20];
        str24 = string[24];
        str30 = string[30];
        trttxt = (nontrtaut, incoherence, ok);
        preptxt = (pret,enprep);
        trtphrase= (incoherente, coherente, nontrt);
        prepphrase = (e1, e2, e3, e4, e5);
(* e1= analyse fournie par le LASLA de Liege                          *)
(* e2= complements manuels de l'analyse en cours                      *)
(* e3= complements manuels de l'analyse termines mais non verifies    *)
(* e4= complements manuels de l'analyse incoherents                   *)
(* e5= analyse coherente, prete                                       *)

        anal_morph_lat = record
                          catgram : char;
                          sscatgram_dgr_vx : char;
                          cas_pers_nbr : char;
                          mode : char;
                          temps : char;
                          fonction : char;
                          emplois : char;
                          genre : char;
                          codesubord : str2;
                          lemme : str16;
                          tradf : str24
                         end;
        list_anal_morph_lat = ^lanalmorphlat;
        lanalmorphlat = record
                         analmorphlat : anal_morph_lat;
                         analmorphlatsuivant : list_anal_morph_lat
                        end;
        list_motlat_analmorph = ^lmotlatanalmorph;
        lmotlatanalmorph = record
                            motlat : str20;
                            analmorphmotlat : list_anal_morph_lat;
                            motlatsuivant : list_motlat_analmorph
                           end;


        listtradfranc = ^ltradfranc;
        ltradfranc = record
                      motlatin : str20;
                      tradfranc : str24;
                      suivant_tf : listtradfranc
                     end;

        typenoeudaa = (n1, n2, n3, n4);
(* n1= choix d'un mot correspondant a une fonction       *)
(* n2= analyse morphologique d'un mot                    *)
(* n3= association d'un ensemble de mots a un symbole categoriel synt.*)
(* n4= traductions admises de tous les mots de la phrase             *)
        typecategorie = (subst, adj, num, adjpron, v, adv, prep, conj, interj);
(* types de cat{gories possibles pour d{crire morphologiquement un mot latin *)
        typeadjpron = (pers, pos, refl, posrefl, dem, relat, inter, ind);
(* types possibles d'un adjectif pronom latin : PERSsonnel, POSsessif, REFL{chi, *)
(* POSsessif REFL{chi, DEMonstratif, RELATif, INTERrogatif ou IND{fini.          *)
        cas = (nom,voc,acc,gen,datif,abl,loc,indecl);
```

```
        nombre = (sing, pluriel,nombrenul);
        genre = (commun, fem, mascfem, masc, mascneutre, neutre,genrenul);
        mode = (indic,imp,subj,part,adjverb,gerondif,inf,supinum,supinu, modenul);
        temps = (pres,impft,futspl,prft,plusqueprft,futant,fuisse, tempsnul);
        degre = (positif, comparatif, superlatif);
        personne = (p1, p2, p3, pnul);
        str80 = string[80];
        type_ib = (indicatif, break);
        arrbool85 = array [1..85] of boolean;
        fonctiontype = (sujet, verbe, cod, cplt, somethingelse);
        declsubsttype = (d1, d2, d3, d4, d5, d6, d7);
        classeadjtype = (a1, a2, a3, a4, a5, a6, a7);
        catnumtype = (card,ord,distr,mult,advord,advmult);
        conjvtype = (c1,c2,c3,c4,c5,c6);
        voixvtype = (actif,passif,deponent,semideponent);
        fonctionvtype = (princ, subord, fonctionnul);
        catadvtype = (rel, int, neg, intneg, comp, superl, catnul);
        caspreptype = (accusatif, genitif, ablatif);
        catconjtype = (coord, sub);
        symbcatsynttype = (gnsujet, gncod, gnabl, whatelse);
        noeudaa = record
                    ind_brk : type_ib;
                    appart_analyse_correct : boolean;
                    comment_syst : str80;
                    comment_specif : str80;
                    accept_comment : boolean;
                    case typenoeud : typenoeudaa of
                        n1 : ( fonction : fonctiontype;
                               mot1 : str20);
                        n2 : ( lemme : str16;
                               case categorie : typecategorie of
                                    subst : ( declsubst : declsubsttype;
                                              cassubst : cas;
                                              nbrsubst : nombre;
                                              genresubst : genre );
                                    adj : ( classeadj : classeadjtype;
                                            degreadj : degre;
                                            casadj : cas ;
                                            nbradj : nombre;
                                            genreadj : genre);
                                    num : ( degrenum : degre;
                                            catnum : catnumtype;
                                            nbrnum : nombre;
                                            casnum : cas;
                                            genrenum : genre );
                                    adjpron : ( catadjpron : typeadjpron;
                                                nbradjpron : nombre;
                                                casadjpron : cas;
                                                genreadjpron : genre;
(* si pronom relatif ou interrogatif *)            modesubadjpron : mode;
(* si pronom relatif ou interrogatif *)            tempssubadjpron : temps );
                                    v : ( conjv : conjvtype;
                                          voixv : voixvtype;
                                          fonctionv : fonctionvtype;
                                          genrev : genre;
                                          tempsv : temps;
                                          case modev : mode of
                                              indic, imp, subj,
                                              inf, supinum, supinu : (persv : personne;
                                                                      nbr1v : nombre);
(* si inf, supinum ou supinu : persv = nbr1v = 0 *)
(* si supinum ou supinu : tempsv = 0 et si inf : tempsv = 1 *)
(* si adjverb ou gerondif : tempsv = 0 et si part : tempsv = 4 ou 1 *)
                                              part, adjverb, gerondif : (casv : cas;
                                                                         nbr2v : nombre) );
```

```
                                adv : (catadv : catadvtype;
(* si interrogatif ou relatif *)          modesubadv : mode;
(* si interrogatif ou relatif *)          tempssubadv : temps );
                                prep : ( typemecum : boolean;
                                         casprep : caspreptype );
                                conj : (catconj : catconjtype;
(* si subordonn{e *)                      modesubconj : mode;
(* si subordonn{e *)                      tempssubconj : temps );
                                interj : ( ) );
                      n3 : ( symbcatsynt : symbcatsynttype;
                             ensmots3 : arrbool85);
   (* 85 = limite maximum de mots dans une phrase (?)   *)
   (* un boolean correspond @ un mot : accept{ ou pas   *)
                      n4 : ( )
                end;
      listnoeudaa = ^lnoeudaa;
      lnoeudaa = record
                   noeud : noeudaa;
                   noeudfils : listnoeudaa;
                   noeudfrere : listnoeudaa;
                   noeudpere : listnoeudaa
                 end;

      fich_cont_txt = text;   (* une ligne = 80 char, longueur r{elle des *)
                              (* mots, point, blanc s{parant les mots et  *)
                              (* nouvelle phrase = nouvelle ligne.        *)
                              (* fichier "fixe" .                         *)

      analmorphmotlat =  record
                           motlat : str20;
                           catgram : char;
                           sscatgram_dgr_vx : char;
                           cas_pers_nbr : char;
                           mode : char;
                           temps : char;
                           fonction : char;
                           emplois : char;
                           genre : char;
                           codesubord : str2;
                           lemme : str16;
                           tradf : str24
                         end;
      fich_analmorph_motlat = file of analmorphmotlat;
                              (* fichier modifiable et valable pour tout texte        *)


      fnoeudaa = record
                   noeud : noeudaa;
                   noeudfils : boolean;
                   noeudfrere : boolean
                 end;
      fich_arbreanalyse = file of fnoeudaa;

      noeudael = record
                   case typenoeudel : typenoeudaa of
                     n1 : ( fonctionel : fonctiontype;
                            motlel : str20);
                     n2 : ( lemmeel : str16;
                            case categorieel : typecategorie of
                                 subst : ( declsubstel : declsubsttype;
                                           cassubstel : cas;
                                           nbrsubstel : nombre;
                                           genresubstel : genre );
                                 adj : ( classeadjel : classeadjtype;
                                         degreadjel : degre;
```

```
                                        casadjel : cas ;
                                        nbradjel : nombre;
                                        genreadjel : genre);
                                num : ( degrenumel : degre;
                                        catnumel : catnumtype;
                                        nbrnumel : nombre;
                                        casnumel : cas;
                                        genrenumel : genre );
                                adjpron : ( catadjpronel : typeadjpron;
                                            nbradjpronel : nombre;
                                            casadjpronel : cas;
                                            genreadjpronel : genre;
(* si pronom relatif ou interrogatif *)       modesubadjpronel : mode;
(* si pronom relatif ou interrogatif *)       tempssubadjpronel : temps );
                                v : ( conjvel : conjvtype;
                                      voixvel : voixvtype;
                                      fonctionvel : fonctionvtype;
                                      genrevel : genre;
                                      tempsvel : temps;
                                      case modevel : mode of
                                          indic, imp, subj,
                                          inf, supinum, supinu : (persvel : personne;
                                                                  nbr1vel : nombre);
(* si inf, supinum ou supinu : persv = nbr1v = 0 *)
(* si supinum ou supinu : tempsv = 0 et si inf : tempsv = 1 *)
(* si adjverb ou gerondif : tempsv = 0 et si part : tempsv = 4 ou 1 *)
                                          part, adjverb, gerondif : (casvel : cas;
                                                                     nbr2vel : nombre) );
                                adv : (catadvel : catadvtype;
(* si interrogatif ou relatif *)            modesubadvel : mode;
(* si interrogatif ou relatif *)            tempssubadvel : temps );
                                prep : ( typemecumel : boolean;
                                         casprepel : caspreptype);
                                conj : (catconjel : catconjtype;
(* si subordonn{e *)                        modesubconjel : mode;
(* si subordonn{e *)                        tempssubconjel : temps );
                                interj : ( ) );
                        n3 : ( symbcatsyntel : symbcatsynttype;
                               ensmots3el : arrbool85);
   (* 85 = limite maximum de mots dans une phrase (?)   *)
   (* un boolean correspond @ un mot : accept{ ou pas   *)
                        n4 : ( )
                  end;
        listnoeudael = ^lnoeudael;
        lnoeudael = record
                      noeudel : noeudael;
                      noeudfilsel : listnoeudael;
                      noeudpereel : listnoeudael
                    end;

        fnoeudael = record
                      noeudel : noeudael;
                      noeudfilsel : boolean
                    end;
        fich_arbreanalyse_el = file of fnoeudael;

        str14 = string[14];
        tdescr = (txtdescr, phrdescr);
        descr_txt = record
                     case typedescr : tdescr of
(* seek (0) *)             txtdescr : (titre_txt : str30;
                                       etat_prep_txt : preptxt;
                                       res_trt_aut : trttxt;
                                       comment_txt : str80;
                                       nomfichcontxt : str14);
```

```
(* seek (nphr) *)        phrdescr : (numero_phrase : integer;
                                    etat_prep_phrase : prepphrase;
                                    res_trt_aut_phrase : trtphrase;
                                    comment_phrase : str80;
                                    nomfichanal : str14)
                          end;
        fich_descr_txt = file of descr_txt;     (* fichier contenant des informations g{n{rales modifiables @ *)
                                                (* propos du texte entier et plus particuli}rement ses phrases.*)

        list_descr = ^descrl;
        descrl = record
                   contenu : descr_txt;
                   suivant : list_descr
                 end;

        indexam = record
                    motlatin : str20;
                    pos : integer
                  end;
        indexfamml = file of indexam;

        lcontphr = ^mcontphr;
        mcontphr = record
                     mot : str20;
                     libre : boolean;
                     suivant : lcontphr
                   end;


  var
        fdescrtxt : fich_descr_txt;     (* pour chaque texte : 'b:*.dsc' *)
        lignefdt : descr_txt;
        fconttxt : fich_cont_txt;       (* pour chaque texte : 'b:*.fct' *)
        lignefconttxt : str80;
        fna : fich_arbreanalyse;        (* pour chaque phrase : 'b:nomfile.nphrase' *)
        fnaligne : fnoeudaa;
        fnael : fich_arbreanalyse_el;   (* pour chaque phrase trait{e par un {l}ve : 'b:nomfile.nphrase'el' *)
        fnaelligne : fnoeudael;
        famml : fich_analmorph_motlat;  (* unique : 'b:fmotlat.am' *)
        lignefamml : analmorphmotlat;
        ligneindex : indexam;
        findexfamml : indexfamml;       (* unique : 'b:index.am' *)


implementation

begin
end.
```

```
Unit LE_UNIT;

Interface
uses ('$U b:globalel ') GLOBALEL, CRT, DOS;

type str60 = string[60];

procedure FIND_INDEX ( var mot : str20; var where : integer);

procedure VISUTXTPHR (var namefct : str14; nphrase : integer; var title : str60;
                      miny : integer; maxy : integer; var result : boolean) ;

procedure VISU_TXT_LAT (var namefcttxt : str14);

procedure AFFICHN3 (var noeudel : noeudael; var firstmcp : lcontphr; miny : integer; maxy : integer);

procedure AFFICHLN3 (var firstln3 : lcontphr; miny : integer; maxy : integer);

procedure REMPLI_N3_LN3 (var firstln3 : lcontphr; var ensmots3 : arrbool85; var firstmcp : lcontphr);

procedure COMPARMOT (var firstlna : listnoeudaa; var lnacour : listnoeudaa; var lnacourel : listnoeudael;
                     var finanbool : boolean; var gotofin : boolean; var firstmcp : lcontphr;
                     var firstlnael : listnoeudael; var existfnael : boolean; var gotodebut : boolean;
                     var namefct : str14; var titretxt : str60; var firstld : list_descr;
                     var ldcour : list_descr; var comment : str80);

procedure COMPARAM (var noeudfilscour : listnoeudaa; var lnacour : listnoeudaa;
                    var noeudfilscourel : listnoeudael; var lnacourel : listnoeudael;
                    categorie : typecategorie; var finanbool : boolean; var gotofin : boolean;
                    var gotoanal : boolean; var existfnael : boolean; var namefct : str14;
                    var titretxt : str60; var firstld : list_descr; ldcour : list_descr);


Implementation


procedure FIND_INDEX;

(* sp(cifications : FIND_INDEX re\oit le mot 'mot' et trouve la position     *)
(*                  'where' de la 1[ analyse morphologique possible de ce mot*)
(*                  dans le fichier des analyses morphologiques.             *)

var min, max, cour : integer;
    ok : boolean;

begin
     assign (findexfamml, 'b:index.am'); reset (findexfamml);
     min := 0; max := ( filesize (findexfamml) - 1 );
     ok := false;
     while ( (ok = false) and (min <= max) ) do
     begin
          cour := (min + max) div 2;
          seek (findexfamml, cour); read (findexfamml, ligneindex);
          if ( ligneindex.motlatin = mot )
          then begin where := ligneindex.pos; ok := true end
          else if ( ligneindex.motlatin < mot )
               then min := cour + 1
               else max := cour - 1
     end;
     close (findexfamml)
end;

(*****************************************************************************)
```

```
procedure VISUTXTPHR;

(* sp(cifications : Si 'nphrase' = 0                                    *)
(*                  Alors VISUTXTPHR affichera @ l'(cran entre les lignes   *)
(*                        'miny' et 'maxy' d'abord 'title', puis toutes les *)
(*                        phrases contenues dans 'namefct'. Si le texte est *)
(*                        trop long que pour apparai^tre en un (cran, la    *)
(*                        pression d'une touche permettra d'en voir la suite.*)
(*                        'result' vaudra alors true.                       *)
(*                  Sinon VISUTXTPHR affichera @ l'(cran entre les lignes   *)
(*                        'miny' et 'maxy' d'abord 'title', puis la phrase  *)
(*                        portant le num(ro 'nphrase' dans le texte 'namefct'*)
(*                        Si la phrase est trop longue que pour apparai^tre en*)
(*                        un (cran, la pression d'une touche permettra d'en *)
(*                        voir la suite.  'result' vaudra true si la phrase *)
(*                        existe dans le texte, sinon il vaudra 'false'.    *)

var i : integer;
    finphrase : boolean;
    ch : char;

begin
      clrscr;
      assign (fconttxt, namefct); reset (fconttxt);
      result := true;
      if (nphrase = 0)
      then begin
                gotoxy (20, miny); writeln (title); writeln;
                while ( not eof(fconttxt) ) do
                begin
                     readln (fconttxt, lignefconttxt);
                     if (wherey > maxy)
                     then begin
                               gotoxy (35, maxy + 2);
                               write ('Tape une touche pour voir la page suivante');
                               repeat until keypressed; ch := readkey;
                               clrscr; gotoxy (20, miny); writeln (title); writeln
                          end;
                     write (lignefconttxt);
                     if (wherex = 1) then gotoxy (80, wherey - 1);
                     writeln
                end
           end
      else begin
                i := 0;
                while ( (not eof(fconttxt)) and (i < nphrase - 1) ) do
                begin
                     readln (fconttxt, lignefconttxt);
                     if (lignefconttxt[length(lignefconttxt)] = '.') then i := i + 1
                end;
                if ( (eof(fconttxt)) and (i <= nphrase - 1) )
                then result := false
                else begin
                          gotoxy (20, miny); writeln (title); writeln;
                          finphrase := false;
                          while ( (not eof(fconttxt)) and (finphrase = false) ) do
                          begin
                               readln (fconttxt, lignefconttxt);
                               if (wherey > maxy)
                               then begin
                                         gotoxy (35, maxy + 2);
                                         write ('Tape une touche pour voir la page suivante');
                                         repeat until keypressed; ch := readkey;
                                         clrscr; gotoxy (20, miny); writeln (title); writeln
                                    end;
```

```
                                write (lignefconttxt);
                                if (wherex = 1) then gotoxy (80, wherey - 1);
                                writeln;
                                if (lignefconttxt[length(lignefconttxt)] = '.') then finphrase := true
                            end
                        end

            end;
        close (fconttxt)
end;

(*************************************************************************)

procedure VISU_TXT_LAT ;

(* sp{cifications : VISU_TXT_LAT permet de visualiser @ l'{cran un texte latin existant. *)
(*                  Si le texte est trop long que pour apparai^tre en un {cran,          *)
(*                  la pression d'une touche au clavier permettra d'en voir la suite .   *)
(*                  De me^me, @ la fin du texte, il suffira d'appuyer sur une touche au  *)
(*                  clavier pour retourner au menu.                                      *)

var ch         : char;
    iocode, k  : integer;
    namefdt    : str14;
    titre      : str60;
    resultat   : boolean;

begin
     k := length(namefcttxt) - 4;
     namefdt := copy (namefcttxt, 1, k);
     namefdt := concat (namefdt, '.dsc');
(*$I-*)
     assign (fdescrtxt, namefdt); reset (fdescrtxt);
     iocode := ioresult;
     if (iocode <> 0)
     then titre := ''
     else begin
               read (fdescrtxt, lignefdt); titre := lignefdt.titre_txt; close (fdescrtxt)
          end;
(*$I+*)
     VISUTXTPHR (namefcttxt, 0, titre, 2, 23, resultat);
     gotoxy (35,25); write ('Tape une touche pour retourner au menu');
     repeat until keypressed; ch := readkey;
end;

(*-------------------------------------------------------------------------*)

procedure AFFICHN3;

(* sp{cifications : sa^chant que 'noeudel' est un noeud d'analyse de type n3, *)
(*                  AFFICHN3 affiche entre les lignes 'miny' et 'maxy',       *)
(*                  les mots correspondant @ 'noeud.ensmots3' @ partir de     *)
(*                  la liste accessible par 'firstmcp'.                       *)

var mcpcour : lcontphr;
    i, longreelle, k : integer;
    ch : char;
    stop : boolean;
    motaecrire : str20;

begin
      clrscr; mcpcour := firstmcp; i := 1;
      while (mcpcour <> nil) do
      begin
           if (noeudel.ensmots3el[i] = true)
```

```
            then begin
                        if (wherey > maxy)
                        then begin
                                    gotoxy (35, maxy + 2);
                                    write ('Tape une touche pour voir la page suivante');
                                    repeat until keypressed; ch := readkey; clrscr
                             end;
                        stop := false; k := 1;
                        repeat
                              if (mcpcour^.mot[k] = ' ')
                              then stop := true
                              else k := k + 1
                        until ( (k > 20) or (stop = true) );
                        longreelle := k - 1; motaecrire := copy (mcpcour^.mot, 1, longreelle);
                        if ((80 - wherex) < longreelle) then writeln;
                        write (motaecrire);
                        if (wherex = 1) then gotoxy (80, wherey - 1)
                                        else begin write (' '); if (wherex = 1) then gotoxy (80, wherey - 1) end
                 end;
            i := i + 1; mcpcour := mcpcour^.suivant
      end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICHLN3;

(* spécifications : AFFICHLN3 affiche entre les lignes 'miny' et 'maxy',     *)
(*                  les mots 'accepté' appartenant @ la liste accessible    *)
(*                  par 'firstmcp'.                                         *)

var ln3cour : lcontphr;
    longreelle, k : integer;
    ch : char;
    stop : boolean;
    motaecrire : str20;

begin
      clrscr; ln3cour := firstln3;
      while (ln3cour <> nil) do
      begin
            if (ln3cour^.libre = true)
            then begin
                        if (wherey > maxy)
                        then begin
                                    gotoxy (35, maxy + 2);
                                    write ('Tape une touche pour voir la page suivante');
                                    repeat until keypressed; ch := readkey; clrscr
                             end;
                        stop := false; k := 1;
                        repeat
                              if (ln3cour^.mot[k] = ' ')
                              then stop := true
                              else k := k + 1
                        until ( (k > 20) or (stop = true) );
                        longreelle := k - 1; motaecrire := copy (ln3cour^.mot, 1, longreelle);
                        if ((80 - wherex) < longreelle) then writeln;
                        write (motaecrire);
                        if (wherex = 1) then gotoxy (80, wherey - 1)
                                        else begin write (' '); if (wherex = 1) then gotoxy (80, wherey - 1) end
                 end;
            ln3cour := ln3cour^.suivant
      end
end;
```

```
(*-------------------------------------------------------------------------*)

procedure REMPLI_N3_LN3;

(* spécifications : REMPLI_N3_LN3 rempli 'ensmots3' @ partir des informations*)
(*                  enregistrées dans la liste accessible par 'firstln3',    *)
(*                  et @ partir de la liste de mots accessible par 'firstmcp'*)

var ln3cour, mcpcour : lcontphr;
    i : integer;

begin
      ln3cour := firstln3;
      for i := 1 to 85 do ensmots3[i] := false;
      while (ln3cour <> nil) do
      begin
           if (ln3cour^.libre = true)
           then begin
                     mcpcour := firstmcp; i := 1;
                     while (mcpcour^.mot <> ln3cour^.mot) do
                     begin mcpcour := mcpcour^.suivant; i := i + 1 end;
                     if (ensmots3[i] = false)
                     then ensmots3[i] := true
                     else begin
                               repeat begin mcpcour := mcpcour^.suivant; i := i + 1 end
                               until ( (mcpcour^.mot = ln3cour^.mot) and (ensmots3[i] = false) );
                               ensmots3[i] := true
                          end
                end;
           ln3cour := ln3cour^.suivant
      end
end;

(*-------------------------------------------------------------------------*)

procedure COMPARMOT;

var trouve, ok, right : boolean;
    ch, ch2 : char;
    mcpcour : lcontphr;
    mememot : str20;

label FIN;

begin
     gotofin := false;
     gotodebut := false;
     lnacour := firstlna;
     trouve := false;
     while ((lnacour <> nil) and (trouve = false)) do
          if (lnacour^.noeud.mot1 = lnacourel^.noeudel.mot1el)
          then trouve := true
          else lnacour := lnacour^.noeudfrere;
     if (trouve = false)
     then
     begin
       gotoxy (1,6);
       writeln ('Erreur ! Tape une touche pour faire une autre proposition ou ESC !');
       write (comment);
       repeat until keypressed;
       ch := readkey;
       if (ch = #27) then begin finanbool := true; gotofin := true; goto FIN end;
       mcpcour := firstmcp;
       while (mcpcour^.mot <> lnacourel^.noeudel.mot1el) do mcpcour := mcpcour^.suivant;
       if (mcpcour^.libre = false)
```

```
  then mcpcour^.libre := true
  else begin
         mememot := mcpcour^.mot;
         repeat mcpcour := mcpcour^.suivant
         until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = false) );
         mcpcour^.libre := true
       end;
  lnacourel := firstlnael;
  existfnael := true;
  gotodebut := true;
  goto FIN
end
else
begin
  if (lnacour^.noeud.accept_comment = true)
  then begin
         gotoxy (1, 6);
         if (lnacour^.noeud.comment_specif <> '')
         then write (lnacour^.noeud.comment_specif)
         else write (lnacour^.noeud.comment_syst)
       end;
  if (lnacour^.noeud.ind_brk = break)
  then begin
         gotoxy (1,9);
         write ('ERREUR IMPARDONNABLE ! Tape une touche pour corriger ton analyse ou ESC !');
         repeat until keypressed;
         ch := readkey;
         if (ch = #27) then begin finanbool := true; gotofin := true; goto FIN end;
         mcpcour := firstmcp;
         while (mcpcour^.mot <> lnacourel^.noeudel.motlel) do mcpcour := mcpcour^.suivant;
         if (mcpcour^.libre = false)
         then mcpcour^.libre := true
         else begin
                mememot := mcpcour^.mot;
                repeat mcpcour := mcpcour^.suivant
                until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = false) );
                mcpcour^.libre := true
              end;
         lnacourel := firstlnael;
         existfnael := true;
         gotodebut := true;
         goto FIN
       end
  else begin (' trouve et indicatif : on continue ! ')
         ok := false;
         while (ok = false) do
         begin
           gotoxy (1,13);
           delline; delline; delline;
           writeln ('Tape F1 pour voir le texte, F2 pour voir le commentaire du texte, F3 pour voir');
           writeln ('le commentaire de la phrase, F4 pour modifier ton choix, ESC pour partir');
           write ('ou une autre touche pour continuer !');
           repeat until keypressed;
           ch := readkey;
           if (ch = #27)
           then begin finanbool := true; ok := true; gotofin := true; goto FIN end
           else if ((ch = #0) and keypressed)
                then begin
                       ch2 := readkey;
                       case ch2 of
                    ('F1')  #59 : VISUTXTPHR (namefct,0,titretxt,1,13,right);
                    ('F2')  #60 : begin
                                    gotoxy(1,13); delline; delline; delline;
                                    write (firstld^.contenu.comment_txt); gotoxy (40, 15);
                                    write ('Tape une touche pour continuer !'); repeat until keypressed; ch := readkey
```

```
                                end;
                        ('F3')  #61 : begin
                                        gotoxy (1, 13); delline; delline; delline;
                                        write (ldcour^.contenu.comment_phrase); gotoxy (40, 15);
                                        write ('Tape une touche pour continuer !'); repeat until keypressed; ch := readkey
                                      end;
                        ('F4')  #62 : begin
                                        ok := true;
                                        mcpcour := firstmcp;
                                        while (mcpcour^.mot <> lnacourel^.noeudel.mot1el) do mcpcour := mcpcour^.suivant;
                                        if (mcpcour^.libre = false)
                                        then mcpcour^.libre := true
                                        else begin
                                               mememot := mcpcour^.mot;
                                               repeat mcpcour := mcpcour^.suivant
                                               until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = false) );
                                               mcpcour^.libre := true
                                             end;
                                        lnacourel := firstlnael;
                                        existfnael := true;
                                        gotodebut := true;
                                        goto FIN
                                      end;
                                else  begin sound (220); delay (200); nosound end
                            end
                          end
                    else ok := true
              end
            end
    end;
FIN :
    write ('')
end;

(*-------------------------------------------------------------------------*)

procedure COMPARAM;

var trouve, ok, right : boolean;
   ch, ch2 : char;

label FIN;

begin
     gotofin := false; gotoanal := false;
     noeudfilscour := lnacour^.noeudfils;
     trouve := false;
     while ((noeudfilscour <> nil) and (trouve = false)) do
     begin
          if (noeudfilscour^.noeud.categorie = noeudfilscourel^.noeudel.categorieel)
          then case noeudfilscour^.noeud.categorie of
                       v : if ((noeudfilscour^.noeud.conjv = noeudfilscourel^.noeudel.conjvel)
                           and (noeudfilscour^.noeud.voixv = noeudfilscourel^.noeudel.voixvel)
                           and (noeudfilscour^.noeud.fonctionv = noeudfilscourel^.noeudel.fonctionvel)
                           and (noeudfilscour^.noeud.genrev = noeudfilscourel^.noeudel.genrevel)
                           and (noeudfilscour^.noeud.tempsv = noeudfilscourel^.noeudel.tempsvel)
                           and (noeudfilscour^.noeud.modev = noeudfilscourel^.noeudel.modevel))
                           then case noeudfilscour^.noeud.modev of
                                    adjverb, part, gerondif :
                                        if ((noeudfilscour^.noeud.casv = noeudfilscourel^.noeudel.casvel)
                                        and (noeudfilscour^.noeud.nbr2v = noeudfilscourel^.noeudel.nbr2vel))
                                        then begin
                                               trouve := true;
                                               lnacour := noeudfilscour;
                                               lnacourel := noeudfilscourel
```

```
                                    end
                                else noeudfilscour := noeudfilscour^.noeudfrere;
                            else if ((noeudfilscour^.noeud.persv = noeudfilscourel^.noeudel.persvel)
                                and (noeudfilscour^.noeud.nbriv = noeudfilscourel^.noeudel.nbrivel))
                                then begin
                                    trouve := true;
                                    lnacour := noeudfilscour;
                                    lnacourel := noeudfilscourel
                                end
                                else noeudfilscour := noeudfilscour^.noeudfrere
                        end
                    else noeudfilscour := noeudfilscour^.noeudfrere;
            subst : if ((noeudfilscour^.noeud.declsubst = noeudfilscourel^.noeudel.declsubstel)
                    and (noeudfilscour^.noeud.cassubst = noeudfilscourel^.noeudel.cassubstel)
                    and (noeudfilscour^.noeud.nbrsubst = noeudfilscourel^.noeudel.nbrsubstel)
                    and (noeudfilscour^.noeud.genresubst = noeudfilscourel^.noeudel.genresubstel))
                    then begin trouve := true; lnacour := noeudfilscour; lnacourel := noeudfilscourel end
                    else noeudfilscour := noeudfilscour^.noeudfrere;
              adj : if ((noeudfilscour^.noeud.classeadj = noeudfilscourel^.noeudel.classeadjel)
                    and (noeudfilscour^.noeud.degreadj = noeudfilscourel^.noeudel.degreadjel)
                    and (noeudfilscour^.noeud.casadj = noeudfilscourel^.noeudel.casadjel)
                    and (noeudfilscour^.noeud.nbradj = noeudfilscourel^.noeudel.nbradjel)
                    and (noeudfilscour^.noeud.genreadj = noeudfilscourel^.noeudel.genreadjel))
                    then begin trouve := true; lnacour := noeudfilscour; lnacourel := noeudfilscourel end
                    else noeudfilscour := noeudfilscour^.noeudfrere;
          adjpron : if ((noeudfilscour^.noeud.catadjpron = noeudfilscourel^.noeudel.catadjpronel)
                    and (noeudfilscour^.noeud.casadjpron = noeudfilscourel^.noeudel.casadjpronel)
                    and (noeudfilscour^.noeud.nbradjpron = noeudfilscourel^.noeudel.nbradjpronel)
                    and (noeudfilscour^.noeud.genreadjpron = noeudfilscourel^.noeudel.genreadjpronel)
                    and (noeudfilscour^.noeud.modesubadjpron = noeudfilscourel^.noeudel.modesubadjpronel)
                    and (noeudfilscour^.noeud.tempssubadjpron=noeudfilscourel^.noeudel.tempssubadjpronel))
                    then begin trouve := true; lnacour := noeudfilscour; lnacourel := noeudfilscourel end
                    else noeudfilscour := noeudfilscour^.noeudfrere;
             prep : if ((noeudfilscour^.noeud.typemecum = noeudfilscourel^.noeudel.typemecumel)
                    and (noeudfilscour^.noeud.casprep = noeudfilscourel^.noeudel.casprepel))
                    then begin trouve := true; lnacour := noeudfilscour; lnacourel := noeudfilscourel end
                    else noeudfilscour := noeudfilscour^.noeudfrere;
             conj : if ((noeudfilscour^.noeud.catconj = noeudfilscourel^.noeudel.catconjel)
                    and (noeudfilscour^.noeud.modesubconj = noeudfilscourel^.noeudel.modesubconjel)
                    and (noeudfilscour^.noeud.tempssubconj = noeudfilscourel^.noeudel.tempssubconjel))
                    then begin trouve := true; lnacour := noeudfilscour; lnacourel := noeudfilscourel end
                    else noeudfilscour := noeudfilscour^.noeudfrere;
        end
    else noeudfilscour := noeudfilscour^.noeudfrere
end;
if (trouve = false)
then
begin
    gotoxy (1,11); delline;
    write ('Erreur ! Tape une touche pour faire une autre proposition  ou ESC !');
    repeat until keypressed;
    ch := readkey;
    if (ch = #27) then begin finanbool := true; gotofin := true; goto FIN end;
    gotoanal := true;
    existfnael := true;
    noeudfilscourel := lnacourel^.noeudfilsel;
    goto FIN
end
else
begin
    if (lnacour^.noeud.accept_comment = true)
    then begin
            gotoxy (1, 11); delline;
            if (lnacour^.noeud.comment_specif <> '')
            then write (lnacour^.noeud.comment_specif)
```

```
                else write (lnacour^.noeud.comment_syst)
            end;
        if (lnacour^.noeud.ind_brk = break)
        then
            begin
              gotoxy (1,13);
              write ('ERREUR IMPARDONNABLE ! Tape une touche pour corriger ton analyse ou ESC !');
              repeat until keypressed;
              ch := readkey;
              if (ch = #27) then begin finanbool := true; gotofin := true; goto FIN end;
              noeudfilscourel := lnacourel;
              lnacourel := lnacourel^.noeudpereel;
              lnacour := lnacour^.noeudpere;
              existfnael := true;
              gotoanal := true;
              goto FIN
            end
        else
            begin (* trouve et indicatif : on continue ! *)
              ok := false;
              while (ok = false) do
              begin
                gotoxy (1,13); delline; delline; delline;
                writeln ('Tape F1 pour voir le texte, F2 pour voir le commentaire du texte, F3 pour voir');
                writeln ('le commentaire de la phrase, F4 pour modifier ton choix, ESC pour partir');
                write ('ou une autre touche pour continuer !');
                repeat until keypressed;
                ch := readkey;
                if (ch = #27)
                then begin finanbool := true; ok := true; gotofin := true; goto FIN end
                else if ((ch = #0) and keypressed)
                     then begin
                            ch2 := readkey;
                            case ch2 of
                          (*F1*)  #59 : VISUTXTPHR (namefct,0,titretxt,1,13,right);
                          (*F2*)  #60 : begin
                                          gotoxy(1,13); delline; delline; delline;
                                          write (firstld^.contenu.comment_txt); gotoxy (40, 15);
                                          write ('Tape une touche pour continuer !'); repeat until keypressed; ch:=readkey
                                        end;
                          (*F3*)  #61 : begin
                                          gotoxy (1, 13); delline; delline; delline;
                                          write (ldcour^.contenu.comment_phrase); gotoxy (40, 15);
                                          write ('Tape une touche pour continuer !'); repeat until keypressed; ch:=readkey
                                        end;
                          (*F4*)  #62 : begin
                                          ok := true;
                                          noeudfilscourel := lnacourel;
                                          lnacourel := lnacourel^.noeudpereel;
                                          lnacour := lnacour^.noeudpere;
                                          existfnael := true;
                                          gotoanal := true;
                                          goto FIN
                                        end;
                                    else  begin sound (220); delay (200); nosound end
                            end
                          end
                     else ok := true
              end
            end
    end;
FIN :
    write ('')
end;
```

```
End.
```

```
Unit CPLTEL;

Interface
uses ('$U b:globalel ') GLOBALEL, CRT, DOS;

procedure REMPLI_AM_SUBST_EL (var noeudel : noeudael; ligne : integer; existam : boolean);

procedure AFFICH_AM_SUBST_EL (var noeudel : noeudael; ligne : integer);

procedure REMPLI_AM_ADJ_EL (var noeudel : noeudael; ligne : integer; existam : boolean);

procedure AFFICH_AM_ADJ_EL (var noeudel : noeudael; ligne : integer);

procedure REMPLI_AM_ADJPRON_EL (var noeudel : noeudael; ligne : integer; existam : boolean);

procedure AFFICH_AM_ADJPRON_EL (var noeudel : noeudael; ligne : integer);

procedure REMPLI_AM_CONJ_EL (var noeudel : noeudael; ligne : integer; existam : boolean);

procedure AFFICH_AM_CONJ_EL (var noeudel : noeudael; ligne : integer);

procedure REMPLI_AM_PREP_EL (var noeudel : noeudael; ligne : integer; existam : boolean);

procedure AFFICH_AM_PREP_EL (var noeudel : noeudael; ligne : integer);

procedure MAJLIBRE (var firstmcp : lcontphr; var noeudel : noeudael; remonte : boolean);

procedure SELECTMOT (var firstmcp : lcontphr; var existfnael : boolean; fonction : fonctiontype;
                     var entete : str30; var lnacourel : listnoeudael; var firstlnael : listnoeudael;
                     var nodepereel : listnoeudael; var remonte : boolean);

procedure REMPLI_AM_V_EL (var noeudel : noeudael; ligne : integer; existam : boolean);

procedure AFFICH_AM_V_EL (var noeudel : noeudael; ligne : integer);

procedure SELECTAM (var entete : str30; var motanalyse : str20; var existfnael : boolean;
                    var noeudfilscourel : listnoeudael; var lnacourel : listnoeudael;
                    var remonte : boolean);

Implementation

procedure REMPLI_AM_SUBST_EL;

var ch : char; ok : boolean ;

begin gotoxy (1, ligne); writeln ('D{clinaison de ce substantif ? ''1''[, ''2''[, ''3''[, ''4''[, ''5''[,');
      write ('anomal (''6'') ou d{cl. gr. (''7'') : _'); gotoxy (35, wherey); ok := false;
      if (existam = true) then case noeudel.declsubstel of
                               d1 : write ('1 ? '); d2 : write ('2 ? '); d3 : write ('3 ? '); d4 : write ('4 ? ');
                               d5 : write ('5 ? '); d6 : write ('6 ? '); d7 : write ('7 ? ') end;
      while (ok = false) do
      if keypressed
      then begin
             ch := readkey;
             case ch of
                  '1' : begin ok := true; gotoxy(1,ligne); insline; write('D{clinaison : 1['); noeudel.declsubstel := d1 end;
                  '2' : begin ok := true; gotoxy(1,ligne); insline; write('D{clinaison : 2['); noeudel.declsubstel := d2 end;
                  '3' : begin ok := true; gotoxy(1,ligne); insline; write('D{clinaison : 3['); noeudel.declsubstel := d3 end;
                  '4' : begin ok := true; gotoxy(1,ligne); insline; write('D{clinaison : 4['); noeudel.declsubstel := d4 end;
                  '5' : begin ok := true; gotoxy(1,ligne); insline; write('D{clinaison : 5['); noeudel.declsubstel := d5 end;
                  '6' : begin ok:=true; gotoxy(1,ligne); insline; write('D{clinaison : anomal'); noeudel.declsubstel:=d6 end;
                  '7' : begin ok:=true; gotoxy(1,ligne);insline;write('D{clinaison : d{cl. gr.');noeudel.declsubstel:=d7 end;
                 #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                            else begin ok:=true;gotoxy (1, ligne);insline;
                                                       case noeudel.declsubstel of
```

```
                                        d1 : write ('D{clinaison : 1['); d2 : write ('D{clinaison : 2[');
                                        d3 : write ('D{clinaison : 3['); d4 : write ('D{clinaison : 4[');
                                        d5 :write('D{clinaison : 5['); d6 :write('D{clinaison : anomal');
                                        d7 : write ('D{clinaison : d{cl. gr.')
                                    end
                                end;
            else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 1); delline; delline;
writeln ('Cas de ce substantif ? nominatif (''1''), vocatif (''2''), accusatif (''3''),');
writeln ('g{nitif (''4''), datif (''5''), ablatif (''6''), locatif (''7'') ');
write ('ou ind{clinable (''8'') : _'); gotoxy (25, wherey); ok := false;
if (existam = true) then case noeudel.cassubstel of
                        nom : write ('1 ? '); voc : write ('2 ? '); acc : write ('3 ? '); gen : write ('4 ? ');
                        datif : write('5 ? '); abl : write('6 ? '); loc : write('7 ? '); indecl : write('8 ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           '1' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : nominatif'); noeudel.cassubstel := nom end;
           '2' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : vocatif'); noeudel.cassubstel := voc end;
           '3' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : accusatif'); noeudel.cassubstel := acc end;
           '4' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : g{nitif'); noeudel.cassubstel := gen end;
           '5' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : datif'); noeudel.cassubstel := datif end;
           '6' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : ablatif'); noeudel.cassubstel := abl end;
           '7' : begin ok := true; gotoxy(1,ligne+1); insline; write('Cas : locatif'); noeudel.cassubstel:= loc end;
           '8' : begin ok:=true; gotoxy(1,ligne+1);insline;write('Cas : ind{clinable'); noeudel.cassubstel:=indecl end;
           #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                      else begin ok:=true;gotoxy (1, ligne+1);insline;
                                           case noeudel.cassubstel of
                                                nom : write ('Cas : nominatif'); voc : write ('Cas : vocatif');
                                                acc : write ('Cas : accusatif'); gen : write ('Cas : g{nitif');
                                                datif :write('Cas : datif');     abl : write('Cas : ablatif');
                                                loc :write('Cas : locatif'); indecl :write('Cas : ind{clinable')
                                           end
                                      end;
           else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 2); delline; delline; delline;
write ('Nombre de ce substantif ? singulier (''S'') ou pluriel (''P'') : _'); gotoxy (62, wherey); ok := false;
if (existam = true) then case noeudel.nbrsubstel of
                        sing : write ('S ? '); pluriel : write ('P ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           'S', 's' : begin ok := true; write (ch); noeudel.nbrsubstel := sing end;
           'P', 'p' : begin ok := true; write (ch); noeudel.nbrsubstel := pluriel end;
           #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                      else begin ok:=true end;
           else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 3); write ('Genre de ce substantif ? commun (''1''), f{minin (''2''), masculin et f{minin (''3''),');
gotoxy (1, ligne + 4); write ('masculin (''4''), masculin et neutre (''5''), neutre (''6'') ou nul (''7'') : _');
gotoxy (71, wherey); ok := false;
if (existam = true) then case noeudel.genresubstel of
                        commun : write ('1 ? '); fem : write ('2 ? '); mascfem : write ('3 ? ');
                        masc : write ('4 ? '); mascneutre : write ('5 ? '); neutre : write ('6 ? ');
                        genrenul : write ('7 ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
```

```
            case ch of
                 '1' : begin ok := true; write (ch); noeudel.genresubstel := commun end;
                 '2' : begin ok := true; write (ch); noeudel.genresubstel := fem end;
                 '3' : begin ok := true; write (ch); noeudel.genresubstel := mascfem end;
                 '4' : begin ok := true; write (ch); noeudel.genresubstel := masc end;
                 '5' : begin ok := true; write (ch); noeudel.genresubstel := mascneutre end;
                 '6' : begin ok := true; write (ch); noeudel.genresubstel := neutre end;
                 '7' : begin ok := true; write (ch); noeudel.genresubstel := genrenul end;
                #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                           else begin ok:=true end;
                 else  begin sound(220); delay (200); nosound end end
            end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_SUBST_EL;

begin gotoxy (1, ligne); write ('D{clinaison de ce substantif : ');
      case noeudel.declsubstel of
           d1 : write ('1['); d2 : write ('2['); d3 : write ('3[');
           d4 : write ('4['); d5 : write ('5['); d6 : write ('anomal'); d7 : write ('d{cl.gr.') end;
      gotoxy (1, ligne+1); write ('Cas : ');
      case noeudel.cassubstel of
           nom   : write ('nominatif'); voc : write ('vocatif'); acc : write ('accusatif'); gen    : write ('g{nitif');
           datif : write ('datif');     abl : write ('ablatif'); loc : write ('locatif');   indecl : write ('ind{clinable')
      end;
      gotoxy (1, ligne+2); write ('Nombre : ');
      case noeudel.nbrsubstel of sing : write ('singulier'); pluriel : write ('pluriel'); nombrenul : write ('-') end;
      gotoxy (1, ligne+3); write ('Genre : ');
      case noeudel.genresubstel of
           commun     : write ('commun');   fem        : write ('f{minin');          mascfem    : write ('masculin-f{minin');
           masc       : write ('masculin'); mascneutre : write ('masculin-neutre');  neutre     : write ('neutre');
           genrenul   : write ('-') end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_ADJ_EL;

var ch : char; ok : boolean ;

begin gotoxy (1, ligne);
      writeln ('Classe de cet adjectif ? ''1''[, 2[-cons (''2''), 2[-er (''3''), 2[-is (''4''),');
      write ('2[-imp. (''5''), anomal (''6'') ou d{cl. gr. (''7'') : _'); gotoxy (50, wherey); ok := false;
      if (existam = true) then case noeudel.classeadjel of
                               a1 : write ('1 ? '); a2 : write ('2 ? '); a3 : write ('3 ? ');
                               a4 : write ('4 ? '); a5 : write ('5 ? '); a6 : write ('6 ? '); a7 : write ('7 ? ') end;
      while (ok = false) do
      if keypressed
      then begin
             ch := readkey;
             case ch of
                  '1' : begin ok := true; gotoxy(1, ligne); insline; write('Classe : 1['); noeudel.classeadjel := a1 end;
                  '2' : begin ok := true; gotoxy(1, ligne); insline; write('Classe : 2[-cons'); noeudel.classeadjel := a2 end;
                  '3' : begin ok := true; gotoxy(1, ligne); insline; write('Classe : 2[-er'); noeudel.classeadjel := a3 end;
                  '4' : begin ok := true; gotoxy(1, ligne); insline; write('Classe : 2[-is'); noeudel.classeadjel := a4 end;
                  '5' : begin ok := true; gotoxy(1,ligne); insline; write('Classe : 2[-imp.'); noeudel.classeadjel := a5 end;
                  '6' : begin ok := true; gotoxy(1,ligne); insline; write('Classe : anomal'); noeudel.classeadjel:=a6 end;
                  '7' : begin ok:=true; gotoxy(1,ligne); insline; write('Classe : d{cl. gr.'); noeudel.classeadjel:=a7 end;
                 #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                            else begin ok:=true;gotoxy (1, ligne);insline;
                                                       case noeudel.classeadjel of
                                                            a1 : write ('Classe : 1['); a2 : write ('Classe : 2[-cons');
                                                            a3 : write ('Classe : 2[-er'); a4 : write ('Classe : 2[-is');
```

```
                                        a5 : write ('Classe : 2[-imp'); a6 : write ('Classe : anomal');
                                        a7 : write ('Classe : d(cl. gr.')
                                    end
                                end;
            else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 1); delline; delline;
write ('Degr( de cet adjectif ? ''P''ositif, ''C''omparatif ou ''S''uperlatif : _'); gotoxy (67, wherey); ok := false;
if (existam = true) then case noeudel.degreadjel of
                        positif : write ('P ? '); comparatif : write ('C ? '); superlatif : write ('S ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           'P', 'p' : begin ok := true; write (ch); noeudel.degreadjel := positif end;
           'C', 'c' : begin ok := true; write (ch); noeudel.degreadjel := comparatif end;
           'S', 's' : begin ok := true; write (ch); noeudel.degreadjel := superlatif end;
               #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                          else begin ok := true end;
           else       begin sound (220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 2);
writeln ('Cas de cet adjectif ? nominatif (''1''), vocatif (''2''), accusatif (''3''),');
writeln ('g(nitif (''4''), datif (''5''), ablatif (''6''), locatif (''7'') ');
write ('ou ind(clinable (''8'') : _'); gotoxy (25, wherey); ok := false;
if (existam = true) then case noeudel.casadjel of
                        nom : write ('1 ? '); voc : write ('2 ? '); acc : write ('3 ? '); gen : write ('4 ? ');
                        datif : write ('5 ? '); abl : write ('6 ? '); loc : write ('7 ? ');
                        indecl : write ('8 ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           '1' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : nominatif'); noeudel.casadjel := nom end;
           '2' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : vocatif'); noeudel.casadjel := voc end;
           '3' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : accusatif'); noeudel.casadjel := acc end;
           '4' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : g(nitif'); noeudel.casadjel := gen end;
           '5' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : datif'); noeudel.casadjel := datif end;
           '6' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : ablatif'); noeudel.casadjel := abl end;
           '7' : begin ok := true; gotoxy(1, ligne+1); insline; write('Cas : locatif'); noeudel.casadjel:= loc end;
           '8' : begin ok:=true; gotoxy(1, ligne+1);insline; write('Cas : ind(clinable'); noeudel.casadjel:=indecl end;
           #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                      else begin ok:=true;gotoxy (1, ligne+1);insline;
                                           case noeudel.casadjel of
                                               nom : write ('Cas : nominatif'); voc : write ('Cas : vocatif');
                                               acc : write ('Cas : accusatif'); gen : write ('Cas : g(nitif');
                                               datif :write('Cas : datif');     abl : write('Cas : ablatif');
                                               loc :write('Cas : locatif'); indecl :write('Cas : ind(clinable')
                                           end
                                end;
            else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 3); delline; delline; delline;
write ('Nombre de cet adjectif ? singulier (''S'') ou pluriel (''P'') : _'); gotoxy (61, wherey); ok := false;
if (existam = true) then case noeudel.nbradjel of
                        sing : write ('S ? '); pluriel : write ('P ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           'S', 's' : begin ok := true; write (ch); noeudel.nbradjel := sing end;
           'P', 'p' : begin ok := true; write (ch); noeudel.nbradjel := pluriel end;
```

```
                    #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                               else begin ok:=true end;
                else  begin sound(220); delay (200); nosound end end
            end;
        gotoxy (1, ligne + 4);
        writeln ('Genre de cet adjectif ? commun (''1''), f{minin (''2''), masculin et f{minin (''3''),');
        write ('masculin (''4''), masculin et neutre (''5'') ou neutre (''6'') : _');
        gotoxy (wherex - 1, wherey); ok := false;
        if (existam = true) then case noeudel.genreadjel of
                                commun : write ('1 ? '); fem : write ('2 ? '); mascfem : write ('3 ? ');
                                masc :write ('4 ? '); mascneutre : write('5 ? '); neutre : write('6 ? ');
                                genrenul : write('7 ? ') end;
        while (ok = false) do
        if keypressed
        then begin
              ch := readkey;
              case ch of
                   '1' : begin ok := true; write (ch); noeudel.genreadjel := commun end;
                   '2' : begin ok := true; write (ch); noeudel.genreadjel := fem end;
                   '3' : begin ok := true; write (ch); noeudel.genreadjel := mascfem end;
                   '4' : begin ok := true; write (ch); noeudel.genreadjel := masc end;
                   '5' : begin ok := true; write (ch); noeudel.genreadjel := mascneutre end;
                   '6' : begin ok := true; write (ch); noeudel.genreadjel := neutre end;
                   #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                              else begin ok:=true end;
                else  begin sound(220); delay (200); nosound end end
          end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_ADJ_EL;

begin gotoxy (1, ligne); write ('Classe de cet adjectif : ');
      case noeudel.classeadjel of
          a1 : write ('1['');      a2 : write ('2[ cons.'); a3 : write ('2[ -er');  a4 : write ('2[ -is');
          a5 : write ('2[ imp.'); a6 : write ('anomal');   a7 : write ('d{cl.gr.') end;
      gotoxy (1, ligne + 1); write ('Degr{ : ');
      case noeudel.degreadjel of
          positif : write ('positif'); comparatif : write ('comparatif'); superlatif : write ('superlatif') end;
      gotoxy (1, ligne+2); write ('Cas : ');
      case noeudel.casadjel of
          nom   : write ('nominatif'); voc : write ('vocatif'); acc : write ('accusatif'); gen    : write ('g{nitif');
          datif : write ('datif');     abl : write ('ablatif'); loc : write ('locatif');   indecl : write ('ind{clinable')
      end;
      gotoxy (40, ligne+2); write ('Nombre : ');
      case noeudel.nbradjel of
          sing      : write ('singulier'); pluriel   : write ('pluriel'); nombrenul : write ('-') end;
      gotoxy (1, ligne+3); write ('Genre : ');
      case noeudel.genreadjel of
          commun   : write ('commun');   fem        : write ('f{minin');        mascfem : write ('masculin-f{minin');
          masc     : write ('masculin'); mascneutre : write ('masculin-neutre'); neutre  : write ('neutre');
          genrenul : write ('-') end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_ADJPRON_EL;

var ch : char; ok : boolean ;

begin gotoxy (1, ligne);
      writeln ('Cat{gorie de cet adjectif pronom ? personnel (''1''), possessif (''2''), ');
      writeln ('r{fl{chi (''3''), possessif r{fl{chi (''4''), d{monstratif (''5''), relatif (''6''),');
      write ('interrogatif (''7'') ou ind{fini (''8'') : _'); gotoxy (40, wherey); ok := false;
```

```
if (existam = true) then case noeudel.catadjpronel of
                        pers : write ('1 ? '); pos : write ('2 ? '); refl : write ('3 ? '); posrefl : write ('4 ? ');
                        dem : write('5 ? '); relat : write('6 ? '); inter : write('7 ? '); ind : write('8 ? ') end;
while (ok = false) do
if keypressed
then begin
       ch := readkey;
       case ch of
         '1' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : personnel'); noeudel.catadjpronel:=pers end;
         '2' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : possessif'); noeudel.catadjpronel:=pos end;
         '3' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : r{fl{chi'); noeudel.catadjpronel:=refl end;
         '4' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : possessif-r{fl{chi');
               noeudel.catadjpronel := posrefl end;
         '5' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : d{monstratif');
               noeudel.catadjpronel:=dem end;
         '6' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : relatif'); noeudel.catadjpronel:= relat end;
         '7' : begin ok:=true; gotoxy(1,ligne); insline; write('Cat{gorie : interrogatif');
               noeudel.catadjpronel:= inter end;
         '8' : begin ok:=true; gotoxy (1,ligne); insline; write ('Cat{gorie : ind{fini'); noeudel.catadjpronel:=ind end;
         #13 : if (existam = false) then begin sound (220); delay (200); nosound end
                                    else begin ok := true; gotoxy (1, ligne); insline;
                                               case noeudel.catadjpronel of
                                                 pers:write('Cat{gorie : personnel');pos:write('Cat{gorie : possessif');
                                              refl:write('Cat{gorie : r{fl{chi');posrefl:write('Cat{gorie : possessif-r{fl{chi');
                                                     dem:write('Cat{gorie : d{monstratif');relat:write('Cat{gorie : relatif');
                                                     inter:write('Cat{gorie : interrogatif');ind:write('Cat{gorie : ind{fini')
                                                     end
                                          end;
        else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 1); delline; delline; delline;
writeln ('Cas de cet adjectif pronom ? nominatif (''1''), vocatif (''2''), accusatif (''3''),');
writeln ('g{nitif (''4''), datif (''5''), ablatif (''6''), locatif (''7'') ');
write ('ou ind{clinable (''8'') : _'); gotoxy (25, wherey); ok := false;
if (existam = true) then case noeudel.casadjpronel of
                        nom : write ('1 ? '); voc : write ('2 ? '); acc : write ('3 ? '); gen : write ('4 ? ');
                        datif : write('5 ? '); abl : write('6 ? '); loc : write('7 ? '); indecl : write('8 ? ') end;
while (ok = false) do
if keypressed
then begin
       ch := readkey;
       case ch of
            '1' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : nominatif'); noeudel.casadjpronel := nom end;
            '2' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : vocatif'); noeudel.casadjpronel := voc end;
            '3' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : accusatif'); noeudel.casadjpronel := acc end;
            '4' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : g{nitif'); noeudel.casadjpronel := gen end;
            '5' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : datif'); noeudel.casadjpronel := datif end;
            '6' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : ablatif'); noeudel.casadjpronel := abl end;
            '7' : begin ok := true; gotoxy(1,ligne+1);insline;write('Cas : locatif'); noeudel.casadjpronel:= loc end;
            '8' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Cas : ind{clinable');noeudel.casadjpronel:=indecl end;
            #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                       else begin ok:=true;gotoxy (1, ligne+1);insline;
                                                  case noeudel.casadjpronel of
                                                       nom : write ('Cas : nominatif'); voc : write ('Cas : vocatif');
                                                       acc : write ('Cas : accusatif'); gen : write ('Cas : g{nitif');
                                                       datif :write('Cas : datif');     abl : write('Cas : ablatif');
                                                       loc :write('Cas : locatif'); indecl :write('Cas : ind{clinable')
                                                  end
                                            end;
           else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 2); delline; delline; delline;
write ('Nombre de cet adjectif pronom ? singulier (''S'') ou pluriel (''P'') : _'); gotoxy (68, wherey); ok := false;
if (existam = true) then case noeudel.nbradjpronel of
                        sing : write ('S ? '); pluriel : write ('P ? ') end;
```

```
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           'S', 's' : begin ok := true; write (ch); noeudel.nbradjpronel := sing end;
           'P', 'p' : begin ok := true; write (ch); noeudel.nbradjpronel := pluriel end;
               #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                          else begin ok:=true end;
          else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 3);
writeln ('Genre de cet adjectif pronom ? commun (''1''), f{minin (''2''), masculin et');
writeln ('f{minin (''3''), masculin (''4''), masculin et neutre (''5''), neutre (''6'')');
write ('ou nul (''7'') : _'); gotoxy (16, wherey); ok := false;
if (existam = true) then case noeudel.genreadjpronel of
                         commun : write ('1 ? '); fem : write ('2 ? '); mascfem : write ('3 ? ');
                         masc :write ('4 ? '); mascneutre : write ('5 ? '); neutre : write ('6 ? ');
                         genrenul : write ('7 ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
           '1' : begin ok:=true;gotoxy(1,ligne+3);insline;write('Genre : commun');noeudel.genreadjpronel:=commun end;
           '2' : begin ok:=true;gotoxy(1,ligne+3);insline;write('Genre : f{minin');noeudel.genreadjpronel:=fem end;
           '3' : begin ok := true; gotoxy(1,ligne+3); insline; write('Genre : masculin-f{minin');
                 noeudel.genreadjpronel := mascfem end;
           '4' : begin ok:=true;gotoxy(1,ligne+3);insline;write('Genre : masculin');noeudel.genreadjpronel:=masc end;
           '5' : begin ok := true; gotoxy(1,ligne+3); insline; write('Genre : masculin-neutre');
                 noeudel.genreadjpronel := mascneutre end;
           '6' : begin ok:=true;gotoxy(1,ligne+3);insline;write('Genre : neutre');noeudel.genreadjpronel:=neutre end;
           '7' : begin ok:=true;gotoxy(1,ligne+3);insline;write('Genre : -');noeudel.genreadjpronel:=genrenul end;
           #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                      else begin ok:=true; gotoxy (1, ligne + 3); insline;
                                           case noeudel.genreadjpronel of
                                                commun : write('Genre : commun'); fem : write('Genre : f{minin');
                                                mascfem : write('Genre : masculin-f{minin');
                                                masc :write('Genre : masculin'); neutre :write('Genre : neutre');
                                                mascneutre : write ('Genre : masculin-neutre');
                                                genrenul : write ('Genre : -') end
                                      end;
          else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 4); delline; delline; delline;
if ( (noeudel.catadjpronel = relat) or (noeudel.catadjpronel = inter) )
then begin
         writeln ('Mode du verbe subordonn( ? Indicatif (''1''), Imp{ratif (''2''), Subjonctif (''3''),');
         writeln ('Participe (''4''), Adjectif verbal (''5''), G{rondif (''6''), Infinitif (''7''),');
         write ('Supin en UM (''8'') ou Supin en U (''9'') ou rien (''0'') : _');
         gotoxy (wherex - 1, wherey); ok := false;
         if (existam = true) then case noeudel.modesubadjpronel of
                                  indic : write ('1 ? '); imp : write ('2 ? '); subj : write ('3 ? ');
                                  part : write ('4 ? '); adjverb : write ('5 ? '); gerondif : write ('6 ? ');
                                  inf : write ('7 ? '); supinum : write ('8 ? '); supinu : write ('9 ? ');
                                  modenul : write ('0 ? ') end;
         while (ok = false) do
         if keypressed
         then begin
               ch := readkey;
               case ch of
                    '1' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn( : indicatif');
                          noeudel.modesubadjpronel := indic end;
                    '2' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn( : imp{ratif');
                          noeudel.modesubadjpronel := imp end;
```

```
        '3' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : subjonctif');
                    noeudel.modesubadjpronel := subj end;
        '4' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : participe');
                    noeudel.modesubadjpronel := part end;
       '5' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : adjectif verbal');
                    noeudel.modesubadjpronel := adjverb end;
        '6' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : g{rondif');
                    noeudel.modesubadjpronel := gerondif end;
        '7' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : infinitif');
                    noeudel.modesubadjpronel := inf end;
        '8' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : supin en UM');
                    noeudel.modesubadjpronel := supinum end;
        '9' : begin ok:=true;gotoxy(1,ligne+4);insline;write('Mode du verbe subordonn{ : supin en U');
                    noeudel.modesubadjpronel := supinu end;
        '0' : begin ok := true; gotoxy (1,ligne+4); insline; write ('Pas de verbe subordonn{ !');
                    noeudel.modesubadjpronel := modenul end;
        #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                   else begin ok:=true; gotoxy (1, ligne + 4); insline;
                                              case noeudel.modesubadjpronel of
                                              indic : write ('Mode du verbe subordonn{ : indicatif');
                                              imp : write ('Mode du verbe subordonn{ : imp{ratif');
                                              subj : write ('Mode du verbe subordonn{ : subjonctif');
                                              part : write ('Mode du verbe subordonn{ : participe');
                                            adjverb : write ('Mode du verbe subordonn{ : adjectif verbal');
                                              gerondif : write ('Mode du verbe subordonn{ : g{rondif');
                                              inf : write ('Mode du verbe subordonn{ : infinitif');
                                              supinum : write ('Mode du verbe subordonn{ : supin en UM');
                                              supinu : write ('Mode du verbe subordonn{ : supin en U');
                                              modenul : write ('Pas de verbe subordonn{') end
                                         end;
      else  begin sound(220); delay (200); nosound end end
end;
gotoxy (1, ligne + 5); delline; delline; delline;
if (noeudel.modesubadjpronel = modenul)
then noeudel.tempssubadjpronel := tempsnul
else
begin
writeln ('Temps de ce verbe subordonn{ ? Pr{sent (''1''),Imparfait (''2''),Futur simple (''3''),');
writeln ('Parfait (''4''), Plus que parfait (''5''), Futur ant{rieur (''6''),');
write ('Fui/fuisse (''7''), Fueram/fuissem (''8''), Fuero/fuisse (''9'') ou Nul (''0'') : _');
gotoxy (wherex - 1, wherey); ok := false;
if (existam = true) then case noeudel.tempssubadjpronel of
                         pres : write ('1 ? '); impft : write ('2 ? '); futspl : write ('3 ? ');
                         prft : write ('4 ? '); plusqueprft : write ('5 ? '); futant : write ('6 ? ');
                         fuisse : write ('7 ? '); tempsnul : write ('0 ? ') end;
while (ok = false) do
if keypressed
then begin
      ch := readkey;
      case ch of
        '1' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : pr{sent');
                    noeudel.tempssubadjpronel := pres end;
        '2' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : imparfait');
                    noeudel.tempssubadjpronel := impft end;
       '3' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : futur simple');
                    noeudel.tempssubadjpronel := futspl end;
        '4' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : parfait');
                    noeudel.tempssubadjpronel := prft end;
        '5' : begin gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : plus que parfait');
                    noeudel.tempssubadjpronel := plusqueprft; ok := true end;
        '6' : begin gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : futur ant{rieur');
                    noeudel.tempssubadjpronel := futant; ok := true end;
        '7' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : ''fuisse''');
                    noeudel.tempssubadjpronel := fuisse end;
        '8' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : ''fuisse''');
```

```
                                noeudel.tempssubadjpronel := fuisse end;
                        '9' : begin ok:=true;gotoxy(1,ligne+5);insline;write('Temps du verbe subordonn{ : ''fuisse''');
                                noeudel.tempssubadjpronel := fuisse end;
                        '0' : begin ok := true; gotoxy(1,ligne+5); insline; write('Temps du verbe subordonn{ : -');
                                noeudel.tempssubadjpronel := tempsnul end;
                        #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                                   else begin ok:=true; gotoxy (1, ligne + 5); insline;
                                                          case noeudel.tempssubadjpronel of
                                                            pres  : write('Temps du verbe subordonn{ : pr{sent');
                                                            impft : write('Temps du verbe subordonn{ : imparfait');
                                                      futspl : write('Temps du verbe subordonn{ : futur simple');
                                                            prft  : write('Temps du verbe subordonn{ : parfait');
                                                   plusqueprft :write('Temps du verbe subordonn{ : plus que parfait');
                                                       futant : write('Temps du verbe subordonn{ : futur ant{rieur');
                                                        fuisse : write('Temps du verbe subordonn{ : fuisse');
                                                      tempsnul : write('Temps du verbe subordonn{ :-') end
                                                    end;
                              else  begin sound(220); delay (200); nosound end end
                          end;
                    gotoxy (1, ligne + 6); delline; delline; delline
                    end
          end
     else begin noeudel.modesubadjpronel := modenul; noeudel.tempssubadjpronel := tempsnul end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_ADJPRON_EL;

begin gotoxy (1, ligne); write ('Cat{gorie de cet adjectif pronom : ');
     case noeudel.catadjpronel of
          pers    : write ('personnel');          pos     : write ('possessif');    refl    : write ('r{fl{chi');
          posrefl : write ('possessif r{fl{chi'); dem     : write ('d{monstratif'); relat   : write ('relatif');
          inter   : write ('interrogatif');       ind     : write ('ind{fini') end;
     gotoxy (1, ligne+1); write ('Cas : ');
     case noeudel.casadjpronel of
          nom   : write ('nominatif'); voc : write ('vocatif'); acc : write ('accusatif'); gen    : write ('g{nitif');
          datif : write ('datif');     abl : write ('ablatif'); loc : write ('locatif');   indecl : write ('ind{clinable')
     end;
     gotoxy (40, ligne+1); write ('Nombre : ');
     case noeudel.nbradjpronel of
          sing      : write ('singulier'); pluriel   : write ('pluriel'); nombrenul : write ('-') end;
     gotoxy (1, ligne+2); write ('Genre : ');
     case noeudel.genreadjpronel of
          commun     : write ('commun');   fem        : write ('f{minin');         mascfem    : write ('masculin-f{minin');
          masc       : write ('masculin'); mascneutre : write ('masculin-neutre'); neutre     : write ('neutre');
          genrenul   : write ('-') end;
     gotoxy (1, ligne+3); write ('Mode Subordonn{ : ');
     case noeudel.modesubadjpronel of
          indic   : write ('indicatif'); imp      : write ('imp{ratif');       subj     : write ('subjonctif');
          part    : write ('participe'); adjverb  : write ('adjectif verbal'); gerondif : write ('g{rondif');
          inf     : write ('infinitif'); supinum :  write ('supin en ''um'''); supinu   : write ('supin en ''u''');
          modenul   : write ('-') end;
     gotoxy (40, ligne+3); write ('Temps Subordonn{ : ');
     case noeudel.tempssubadjpronel of
          pres   : write ('pr{sent');    impft       : write ('imparfait');       futspl : write ('futur simple');
          prft   : write ('parfait');    plusqueprft : write ('plus que parfait'); futant : write ('futur ant{rieur');
          fuisse : write ('''fuisse'''); tempsnul    : write ('-') end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_CONJ_EL;

var ch : char; ok : boolean ;
```

```
begin gotoxy (1, ligne);
      write ('Cat{gorie de cette conjonction ? ''C''oordination ou ''S''ubordination : _'); gotoxy (70, wherey); ok := false;
      if (existam = true) then case noeudel.catconjel of
                                  coord : write ('C ? '); sub : write ('S ? ') end;
      while (ok = false) do
      if keypressed
      then begin
            ch := readkey;
            case ch of
                 'C', 'c' : begin ok := true; write (ch); noeudel.catconjel := coord end;
                 'S', 's' : begin ok := true; write (ch); noeudel.catconjel := sub end;
                     #13 : if (existam = false) then begin sound(220); delay (200); nosound end
                                                else ok := true ;
                 else      begin sound (220); delay (200); nosound end end
          end;
      if (noeudel.catconjel = coord)
      then begin noeudel.modesubconjel := modenul; noeudel.tempssubconjel := tempsnul end
      else begin
               gotoxy (1, ligne + 1);
               writeln ('Mode du verbe subordonn{ ? Indicatif (''1''), Imp{ratif (''2''), Subjonctif (''3''),');
               writeln ('Participe (''4''), Adjectif verbal (''5''), G{rondif (''6''), Infinitif (''7''),');
               write ('Supin en UM (''8'') ou Supin en U (''9'') ou rien (''0'') : _');
               gotoxy (wherex - 1, wherey); ok := false;
               if (existam = true) then case noeudel.modesubconjel of
                                           indic : write ('1 ? '); imp : write ('2 ? '); subj : write ('3 ? ');
                                           part : write ('4 ? '); adjverb : write ('5 ? '); gerondif : write ('6 ? ');
                                           inf : write ('7 ? '); supinum : write ('8 ? '); supinu : write ('9 ? ');
                                           modenul : write ('0 ? ') end;
               while (ok = false) do
               if (keypressed)
               then begin
                      ch := readkey;
                      case ch of
                         '1' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : indicatif');
                                     noeudel.modesubconjel := indic end;
                         '2' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : imp{ratif');
                                     noeudel.modesubconjel := imp end;
                         '3' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : subjonctif');
                                     noeudel.modesubconjel := subj end;
                         '4' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : participe');
                                     noeudel.modesubconjel := part end;
                         '5' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : adjectif verbal');
                                     noeudel.modesubconjel := adjverb end;
                         '6' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : g{rondif');
                                     noeudel.modesubconjel := gerondif end;
                         '7' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : infinitif');
                                     noeudel.modesubconjel := inf end;
                         '8' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : supin en UM');
                                     noeudel.modesubconjel := supinum end;
                         '9' : begin ok:=true;gotoxy(1,ligne+1);insline;write('Mode du verbe subordonn{ : supin en U');
                                     noeudel.modesubconjel := supinu end;
                         '0' : begin ok:=true;gotoxy(1,ligne+1);insline;write ('Pas de verbe subordonn{ !');
                                     noeudel.modesubconjel := modenul end;
                         #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                                    else begin ok:=true; gotoxy (1, ligne + 4); insline;
                                                               case noeudel.modesubconjel of
                                                                 indic : write('Mode du verbe subordonn{ : indicatif');
                                                                 imp : write ('Mode du verbe subordonn{ : imp{ratif');
                                                                 subj : write ('Mode du verbe subordonn{ : subjonctif');
                                                                 part : write ('Mode du verbe subordonn{ : participe');
                                                               adjverb : write ('Mode du verbe subordonn{ : adjectif verbal');
                                                                 gerondif : write ('Mode du verbe subordonn{ : g{rondif');
                                                                 inf : write ('Mode du verbe subordonn{ : infinitif');
                                                                 supinum : write ('Mode du verbe subordonn{ : supin en UM');
```

```
                                                    supinu : write ('Mode du verbe subordonn{ : supin en U');
                                                    modenul : write ('Mode du verbe subordonn{ : -') end
                                                end;
                    else  begin sound(220); delay (200); nosound end end
                end;
                gotoxy (1, ligne + 2); delline; delline; delline;
                if (noeudel.modesubconjel = modenul)
                then noeudel.tempssubconjel := tempsnul
                else
                begin
                writeln ('Temps de ce verbe subordonn{ ? Pr{sent (''1''),Imparfait (''2''),Futur simple (''3''),');
                writeln ('Parfait (''4''), Plus que parfait (''5''), Futur ant{rieur (''6''),');
                write ('Fui/fuisse (''7''), Fueram/fuissem (''8''), Fuero/fuisse (''9'') ou Nul (''0'') : _');
                gotoxy (wherex - 1, wherey); ok := false;
                if (existam = true) then case noeudel.tempssubconjel of
                                        pres : write ('1 ? '); impft : write ('2 ? '); futspl : write ('3 ? ');
                                        prft : write ('4 ? '); plusqueprft : write ('5 ? '); futant : write ('6 ? ');
                                        fuisse : write ('7 ? '); tempsnul : write ('0 ? ') end;
                while (ok = false) do
                if keypressed
                then begin
                      ch := readkey;
                      case ch of
                        '1' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Temps du verbe subordonn{ : pr{sent');
                                    noeudel.tempssubconjel := pres end;
                        '2' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : imparfait');
                                    noeudel.tempssubconjel := impft end;
                       '3' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : futur simple');
                                    noeudel.tempssubconjel := futspl end;
                        '4' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : parfait');
                                    noeudel.tempssubconjel := prft end;
                        '5' : begin gotoxy(1,ligne+2);insline;write('Temps du verbe subordonn{ : plus que parfait');
                                    noeudel.tempssubconjel := plusqueprft; ok := true end;
                        '6' : begin gotoxy(1,ligne+2);insline;write('Temps du verbe subordonn{ : futur ant{rieur');
                                    noeudel.tempssubconjel := futant; ok := true end;
                        '7' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : ''fuisse''');
                                    noeudel.tempssubconjel := fuisse end;
                        '8' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : ''fuisse''');
                                    noeudel.tempssubconjel := fuisse end;
                        '9' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : ''fuisse''');
                                    noeudel.tempssubconjel := fuisse end;
                        '0' : begin ok:=true;gotoxy(1,ligne+2);insline; write('Temps du verbe subordonn{ : -');
                                    noeudel.tempssubconjel := tempsnul end;
                        #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                                   else begin ok:=true; gotoxy (1, ligne + 5); insline;
                                                              case noeudel.tempssubconjel of
                                                                 pres  : write('Temps du verbe subordonn{ : pr{sent');
                                                                 impft : write('Temps du verbe subordonn{ : imparfait');
                                                              futspl : write('Temps du verbe subordonn{ : futur simple');
                                                                 prft  : write('Temps du verbe subordonn{ : parfait');
                                                           plusqueprft :write('Temps du verbe subordonn{ : plus que parfait');
                                                              futant : write('Temps du verbe subordonn{ : futur ant{rieur');
                                                                  fuisse : write('Temps du verbe subordonn{ : fuisse');
                                                                tempsnul : write('Temps du verbe subordonn{ : -') end
                                                    end;
                          else  begin sound(220); delay (200); nosound end end
                      end;
                gotoxy (1, ligne + 3); delline; delline; delline
                end
        end
end;

{*-------------------------------------------------------------------------*}

procedure AFFICH_AM_CONJ_EL;
```

```
begin gotoxy (1, ligne); write ('Cat{gorie de cette conjonction : ');
      case noeudel.catconjel of coord : write ('coordination'); sub : write ('subordination') end;
      gotoxy (1, ligne+1); write ('Mode Subordonn{ : ');
      case noeudel.modesubconjel of
           indic   : write ('indicatif'); imp     : write ('imp{ratif');       subj     : write ('subjonctif');
           part    : write ('participe'); adjverb : write ('adjectif verbal'); gerondif : write ('g{rondif');
           inf     : write ('infinitif'); supinum : write ('supin en ''um'''); supinu   : write ('supin en ''u''');
           modenul : write ('-') end;
      gotoxy (1, ligne+2); write ('Temps Subordonn{ : ');
      case noeudel.tempssubconjel of
           pres   : write ('pr{sent');    impft        : write ('imparfait');        futspl : write ('futur simple');
           prft   : write ('parfait');    plusqueprft  : write ('plus que parfait'); futant : write ('futur ant{rieur');
           fuisse : write ('''fuisse'''); tempsnul     : write ('-') end
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_PREP_EL;

var ch : char; ok : boolean ;

begin gotoxy (1, ligne); write ('Type de cette pr{position : MECUM ? ''O''ui ou ''N''on : _');
      gotoxy (54, wherey); ok := false;
      if (existam = true) then if (noeudel.typemecumel = true) then write ('O ? ') else write ('N ? ');
      while (ok = false) do
      if keypressed
      then begin
             ch := readkey;
             case ch of
                  'O', 'o' : begin ok := true; write (ch); noeudel.typemecumel := true end;
                  'N', 'n' : begin ok := true; write (ch); noeudel.typemecumel := false end;
                      #13 : if (existam = false) then begin sound (220); delay (200); nosound end
                                                 else ok := true;
                  else      begin sound (220); delay (200); nosound end end
          end;
      gotoxy (1, ligne + 1);
      writeln ('Cas r{gi par cette pr{position ? accusatif (''1''), g{nitif (''2'') ');
      write ('ou ablatif (''3'') : _'); gotoxy (20, wherey); ok := false;
      if (existam = true) then case noeudel.casprepel of
                               accusatif : write ('1 ? '); genitif : write ('2 ? '); ablatif : write ('3 ? ') end;
      while (ok = false) do
      if keypressed
      then begin
             ch := readkey;
             case ch of
                  '1' : begin ok := true; write (ch); noeudel.casprepel := accusatif end;
                  '2' : begin ok := true; write (ch); noeudel.casprepel := genitif end;
                  '3' : begin ok := true; write (ch); noeudel.casprepel := ablatif end;
                  #13 : if (existam = false) then begin sound (220); delay (200); nosound end
                                             else ok := true;
                  else  begin sound(220); delay (200); nosound end end
          end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_PREP_EL;

begin gotoxy (1, ligne); write ('Type MECUM ? : ');
      case noeudel.typemecumel of true : write ('oui'); false : write ('non') end;
      gotoxy (1, ligne+1); write ('Cas r{gi par cette pr{position : ');
      case noeudel.casprepel of accusatif : write ('accusatif'); genitif : write ('g{nitif'); ablatif : write ('ablatif') end
end;
```

```
(*-------------------------------------------------------------------------*)

procedure MAJLIBRE;

var mcpcour : lcontphr; mememot : str20;

begin mcpcour := firstmcp;
      while (mcpcour^.mot <> noeudel.motlel) do mcpcour := mcpcour^.suivant;
      if (remonte = false)
      then if (mcpcour^.libre = true)
           then mcpcour^.libre := false
           else begin
                  mememot := mcpcour^.mot;
                  repeat mcpcour := mcpcour^.suivant until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = true) );
                  mcpcour^.libre := false
                end
      else if (mcpcour^.libre = false)
           then mcpcour^.libre := true
           else begin
                  mememot := mcpcour^.mot;
                  repeat mcpcour := mcpcour^.suivant until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = false) );
                  mcpcour^.libre := true
                end
end;

(*$I b:cpltpr2.pas *)

End.
```

```
procedure SELECTMOT;

var selection, ok, oke : boolean;
    mcpcour : lcontphr;
    newlnael : listnoeudael;
    ch, ch2 : char;

label FIN;

begin
     remonte := false;
     selection := false; mcpcour := firstmcp;
     while (selection = false) do
     begin
       if (existfnael = false)
       then
       begin
         selection := true; new (newlnael); newlnael^.noeudfilsel := nil;
         newlnael^.noeudpereel := nodepereel;
         newlnael^.noeudel.typenoeudel := ni; newlnael^.noeudel.fonctionel := fonction;
         clrscr; ok := false;
         while (ok = false) do
         begin
           if (mcpcour^.libre = true)
           then begin
                  gotoxy (1, 2); delline; write (entete, mcpcour^.mot, ' ? ''O''-''N'' ou Remont{e-''F5'' :_');
                  gotoxy (wherex -1, wherey); oke := false;
                  while (oke = false) do
                  if keypressed
                  then begin
                         ch := readkey;
                         case ch of
                              'O', 'o' : begin oke := true; ok := true; mcpcour^.libre := false;
                                               newlnael^.noeudel.motiel := mcpcour^.mot; write (ch) end;
                              'N', 'n' : begin oke := true; write (ch) end;
                                  #0   : if keypressed
                                         then begin
                                                ch2 := readkey;
                                                if (ch2 = #63) then begin ok := true; remonte := true; goto FIN end
                                                               else begin sound (220); delay (200); nosound end
                                              end
                                         else begin sound (220); delay (200); nosound end;
                                  else   begin sound (220); delay (200); nosound end end
                       end;
                if (ok = false) then begin mcpcour := mcpcour^.suivant; if (mcpcour = nil) then mcpcour := firstmcp end
              end
           else begin mcpcour := mcpcour^.suivant ; if (mcpcour = nil) then mcpcour := firstmcp end
         end;
         firstlnael := newlnael; lnacourel := newlnael
       end
       else
       begin
         clrscr; gotoxy (1, 2); delline; write (entete, lnacourel^.noeudel.motiel);
         gotoxy (1, 5); write ('S{lection ? ''O''-''N'' ou Remont{e ''F5'' : _'); gotoxy (wherex - 1, wherey); ok := false;
         while (ok = false) do
         begin
           if keypressed
           then begin
                  ch := readkey;
                  case ch of
                       'O', 'o' : begin ok := true; selection := true; write (ch);
                                        MAJLIBRE (firstmcp, lnacourel^.noeudel, false) end;
                       'N', 'n' : begin
                                    ok := true; write (ch);
                                    existfnael := false
```

```
                              end;
                         #0  : if keypressed then begin
                                                    ch2 := readkey;
                                                    if (ch2 = #63) then begin ok := true; remonte := true; goto FIN end
                                                                   else begin sound (220); delay (200); nosound end
                                                  end
                                                else begin sound (220); delay (200); nosound end;
                    else       begin sound (220); delay (200); nosound end end
               end
          end
         end
       end;
FIN :
     write ('')
end;

(*-------------------------------------------------------------------------*)

procedure REMPLI_AM_V_EL;

var ok : boolean; ch : char;

begin gotoxy (1, ligne); write ('Conjugaison de ce verbe ? ''1'', ''2'', ''3'', ''4'', ''5'' ou anomal ''6'' : _');
      gotoxy (wherex - 1, wherey); ok := false;
      if (existam = true) then case noeudel.conjvel of
                                    c1 : write ('1 ? '); c2 : write ('2 ? '); c3 : write ('3 ? '); c4 : write ('4 ? ');
                                    c5 : write ('5 ? '); c6 : write ('6 ? ') end;
      while (ok = false) do
      if (keypressed) then begin
                           ch := readkey;
                           case ch of
                                '1' : begin write (ch); ok := true; noeudel.conjvel := c1 end;
                                '2' : begin write (ch); ok := true; noeudel.conjvel := c2 end;
                                '3' : begin write (ch); ok := true; noeudel.conjvel := c3 end;
                                '4' : begin write (ch); ok := true; noeudel.conjvel := c4 end;
                                '5' : begin write (ch); ok := true; noeudel.conjvel := c5 end;
                                '6' : begin write (ch); ok := true; noeudel.conjvel := c6 end;
                                #13 : if (existam = false) then begin sound (220); delay (200); nosound end
                                                           else ok := true;
                                else  begin sound(220); delay (200); nosound end end
                          end;
      gotoxy (1, ligne + 1); write ('Voix de ce verbe ? ''A''ctif, ''P''assif, ''D''{ponent, ''S''emi-d{ponent : _');
      gotoxy (wherex - 1, wherey); ok := false;
      if (existam = true) then case noeudel.voixvel of
                                    actif : write ('A ? '); passif : write ('P ? '); deponent : write ('D ? ');
                                    semideponent : write ('S ? ') end;
      while (ok = false) do
      if (keypressed) then begin
                           ch := readkey;
                           case ch of
                                'A', 'a' : begin write (ch); ok := true; noeudel.voixvel := actif end;
                                'P', 'p' : begin write (ch); ok := true; noeudel.voixvel := passif end;
                                'D', 'd' : begin write (ch); ok := true; noeudel.voixvel := deponent end;
                                'S', 's' : begin write (ch); ok := true; noeudel.voixvel := semideponent end;
                                #13 : if (existam = false) then begin sound (220); delay (200); nosound end
                                                           else ok := true;
                                else       begin sound (220); delay (200); nosound end end
                          end;
      gotoxy (1, ligne + 2);
      writeln ('Mode de ce verbe ? Indicatif (''1''), Imp{ratif (''2''), Subjonctif (''3''),');
      writeln ('Participe (''4''), Adjectif verbal (''5''), G{rondif (''6''), Infinitif (''7''),');
      write ('Supin en UM (''8'') ou Supin en U (''9'') : _'); gotoxy (wherex - 1, wherey); ok := false;
      if (existam = true) then case noeudel.modevel of
                                    indic : write ('1 ? '); imp : write ('2 ? '); subj : write ('3 ? ');
                                    part : write ('4 ? '); adjverb : write ('5 ? '); gerondif : write('6 ? ');
```

```
                                inf : write('7 ? '); supinum : write('8 ? '); supinu : write('9 ? ');
                                modenul : write('0 ? ') end;
while (ok = false) do
if (keypressed)
then begin
       ch := readkey;
       case ch of
       '1' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : indicatif'); noeudel.modevel:=indic end;
       '2' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : imp{ratif'); noeudel.modevel:=imp end;
       '3' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : subjonctif'); noeudel.modevel:=subj end;
       '4' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : participe'); noeudel.modevel:=part end;
'5' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : adjectif verbal');noeudel.modevel:=adjverb end;
  '6' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : g{rondif');noeudel.modevel:=gerondif end;
       '7' : begin ok:=true;gotoxy(1,ligne+2);insline;write ('Mode de ce verbe : infinitif');noeudel.modevel:=inf end;
  '8' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : supin en UM');noeudel.modevel:=supinum end;
  '9' : begin ok:=true;gotoxy(1,ligne+2);insline;write('Mode de ce verbe : supin en U');noeudel.modevel:=supinu end;
      #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                              else begin ok:=true; gotoxy (1, ligne + 2); insline;
                                      case noeudel.modevel of
                                           indic : write('Mode de ce verbe : indicatif');
                                           imp : write ('Mode de ce verbe : imp{ratif');
                                           subj : write ('Mode de ce verbe : subjonctif');
                                           part : write ('Mode de ce verbe : participe');
                                           adjverb : write ('Mode de ce verbe : adjectif verbal');
                                           gerondif : write ('Mode de ce verbe : g{rondif');
                                           inf : write ('Mode de ce verbe : infinitif');
                                           supinum : write ('Mode de ce verbe : supin en UM');
                                           supinu : write ('Mode de ce verbe : supin en U');
                                           modenul : write ('Pas de verbe subordonn{') end
                                     end;
      else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 3); delline; delline; delline;
writeln ('Temps de ce verbe ? Pr{sent (''1''), Imparfait (''2''), Futur simple (''3''),');
writeln ('Parfait (''4''), Plus que parfait (''5''), Futur ant{rieur (''6''),');
write ('Fui/fuisse (''7''), Fueram/fuissem (''8''), Fuero/fuisse (''9'') ou Nul (''0'') : _');
gotoxy (wherex - 1, wherey); ok := false;
if (existam = true) then case noeudel.tempsvel of
                               pres : write ('1 ? '); impft : write ('2 ? '); futspl : write ('3 ? ');
                               prft : write ('4 ? '); plusqueprft : write ('5 ? '); futant : write('6 ? ');
                               fuisse : write('7 ? '); tempsnul : write('0 ? ') end;
while (ok = false) do
if (keypressed)
then begin
       ch := readkey;
       case ch of
            '1' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : pr{sent');
                        noeudel.tempsvel := pres end;
            '2' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : imparfait');
                        noeudel.tempsvel := impft end;
            '3' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : futur simple');
                        noeudel.tempsvel := futspl end;
            '4' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : parfait');
                        noeudel.tempsvel := prft end;
            '5' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : plus que parfait');
                        noeudel.tempsvel := plusqueprft end;
            '6' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : futur ant{rieur');
                        noeudel.tempsvel := futant end;
            '7' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : ''fuisse''');
                        noeudel.tempsvel := fuisse end;
            '8' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : ''fuisse''');
                        noeudel.tempsvel := fuisse end;
            '9' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : ''fuisse''');
                        noeudel.tempsvel := fuisse end;
            '0' : begin ok := true; gotoxy (1, ligne + 3); insline; write ('Temps de ce verbe : -');
```

```
                    noeudel.tempsvel := tempsnul end;
            #13 : if (existam = false) then begin sound(220); delay(200); nosound end
                                       else begin ok:=true; gotoxy (1, ligne + 3); insline;
                                                  case noeudel.tempsvel of
                                                      pres : write('Temps de ce verbe : pr{sent');
                                                      impft : write ('Temps de ce verbe : imparfait');
                                                      futspl : write ('Temps de ce verbe : futur simple');
                                                      prft : write ('Temps de ce verbe : parfait');
                                                      plusqueprft : write('Temps de ce verbe : plus que parfait');
                                                      futant : write('Temps de ce verbe : futur ant{rieur');
                                                      fuisse : write ('Temps de ce verbe : fuisse');
                                                      tempsnul : write ('Temps de ce verbe :-') end
                                                 end;
            else  begin sound(220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 4); delline; delline; delline;
case noeudel.modevel of
    indic, imp, subj, inf, supinum, supinu : begin
                                                write ('Personne ? ''1''[, ''2''[, ''3''[ ou ''N''ulle : _');
                                                gotoxy (wherex - 1, wherey); ok := false;
                                                if (existam = true)
                                                then case noeudel.persvel of
                                                      p1 : write('1 ? '); p2 : write ('2 ? '); p3 : write ('3 ? ');
                                                      pnul : write ('- ? ') end;
                                                while (ok = false) do
                                                if (keypressed)
                                                then begin
                                                      ch := readkey;
                                                      case ch of
                                                      '1' : begin gotoxy(1,ligne+4);insline;write('Personne : 1[');
                                                            ok:=true; noeudel.persvel:=p1 end;
                                                      '2' : begin gotoxy(1,ligne+4);insline;write('Personne : 2[');
                                                            ok:=true; noeudel.persvel:=p2 end;
                                                      '3' : begin gotoxy(1,ligne+4);insline;write('Personne : 3[');
                                                            ok:=true; noeudel.persvel:=p3 end;
                                                    'N', 'n' : begin gotoxy(1,ligne+4);insline;write('Personne : -');
                                                            ok:=true; noeudel.persvel:=pnul end;
                                                      #13 : if (existam = false)
                                                            then begin sound(220); delay(200); nosound end
                                                            else begin ok := true; gotoxy (1,ligne+4); insline;
                                                                  case noeudel.persvel of
                                                                      p1 : write('Personne : 1[');
                                                                      p2 : write ('Personne : 2[');
                                                                      p3 : write ('Personne : 3[');
                                                                      pnul : write ('Personne : -') end
                                                                 end;
                                                      else  begin sound (220); delay (200); nosound end end
                                                     end;
                                                gotoxy (1, ligne + 5); delline;
                                                write ('Nombre ? ''S''ingulier ou ''P''luriel ou ''N''ul : _');
                                                gotoxy (wherex - 1, wherey); ok := false;
                                                if (existam = true)
                                                then case noeudel.nbrivel of
                                                      sing : write('S ? '); pluriel : write ('P ? ');
                                                      nombrenul : write ('N ? ') end;
                                                while (ok = false) do
                                                if (keypressed)
                                                then begin
                                                      ch := readkey;
                                                      case ch of
                                                      'S', 's' : begin ok := true; noeudel.nbrivel := sing;
                                                            gotoxy(40,ligne+4);write('Nombre : singulier')end;
                                                      'P', 'p' : begin ok := true; noeudel.nbrivel := pluriel;
                                                            gotoxy(40,ligne+4);write('Nombre : pluriel') end;
                                                      'N', 'n' : begin ok := true; noeudel.nbrivel := nombrenul;
```

```
                                        gotoxy(40,ligne+4); write('Nombre : -') end;
                                    #13 : if (existam = false)
                                          then begin sound(220); delay(200); nosound end
                                          else begin ok := true; gotoxy(1,ligne+4); insline;
                                                     case noeudel.nbr1vel of
                                                     sing : write('Nombre : singulier');
                                                     pluriel : write ('Nombre : pluriel');
                                                     nombrenul : write ('Nombre : -') end
                                               end;
                                    else       begin sound (220); delay (200); nosound end end
                                end;
                            gotoxy (1, ligne + 5); delline
                        end;
part, adjverb, gerondif :   begin
                writeln ('Cas ? Nominatif (''1''), Vocatif (''2''), Accusatif (''3''), G{nitif (''4''),');
                write ('Datif (''5''), Ablatif (''6''), Locatif (''7''), Ind{clinable (''8'') : _');
                            gotoxy (wherex - 1, wherey); ok := false;
                            if (existam = true)
                            then case noeudel.casvel of
                                  nom : write('1 ? '); voc : write('2 ? '); acc : write('3 ?');
                                  gen : write ('4 ? '); datif : write ('5 ? ');
                                  abl : write ('6 ? '); loc : write ('7 ? ');
                                  indecl : write ('8 ? ') end;
                            while (ok = false) do
                            if (keypressed)
                            then begin
                                  ch := readkey;
                                  case ch of
                                  '1' : begin ok:=true; noeudel.casvel:=nom; gotoxy(1,ligne+4);
                                              insline; write('Cas : nominatif') end;
                                  '2' : begin ok:=true; noeudel.casvel:=voc; gotoxy(1,ligne+4);
                                              insline; write('Cas : vocatif') end;
                                  '3' : begin ok:=true; noeudel.casvel:=acc; gotoxy(1,ligne+4);
                                              insline; write('Cas : accusatif') end;
                                  '4' : begin ok:=true; noeudel.casvel:=gen; gotoxy(1,ligne+4);
                                              insline; write('Cas : g{nitif') end;
                                  '5' : begin ok:=true;noeudel.casvel:=datif;gotoxy(1,ligne+4);
                                              insline; write('Cas : datif') end;
                                  '6' : begin ok:=true; noeudel.casvel:=abl; gotoxy(1,ligne+4);
                                              insline; write('Cas : ablatif') end;
                                  '7' : begin ok:=true; noeudel.casvel:=loc; gotoxy(1,ligne+4);
                                              insline; write('Cas : locatif') end;
                                  '8': begin ok:=true;noeudel.casvel:=indecl;gotoxy(1,ligne+4);
                                              insline; write('Cas : ind{clinable')end;
                                 #13 : if (existam = false)
                                       then begin sound(220); delay(200); nosound end
                                       else begin ok := true; gotoxy(1,ligne+4); insline;
                                                  case noeudel.casvel of
                                                  nom : write('Cas : nominatif');
                                                  voc : write ('Cas : vocatif');
                                                  acc : write ('Cas : accusatif');
                                                  gen : write ('Cas : g{nitif');
                                                  datif : write ('Cas : datif');
                                                  abl : write ('Cas : ablatif');
                                                  loc : write ('Cas : locatif');
                                                  indecl : write ('Cas : ind{clinable') end
                                            end;
                                  else  begin sound (220); delay (200); nosound end end
                                end;
                            gotoxy (1, ligne + 5); delline; delline;
                            write ('Nombre ? ''S''ingulier ou ''P''luriel : _');
                            gotoxy (wherex - 1, wherey); ok := false;
                            if (existam = true)
                            then case noeudel.nbr2vel of
                                  sing : write('S ? '); pluriel : write ('P ? ') end;
```

```
                                        while (ok = false) do
                                        if (keypressed)
                                        then begin
                                              ch := readkey;
                                              case ch of
                                              'S', 's' : begin ok := true; noeudel.nbr2vel := sing;
                                                         gotoxy(40,ligne+4);write('Nombre : singulier')end;
                                              'P', 'p' : begin ok := true; noeudel.nbr2vel := pluriel;
                                                         gotoxy(40,ligne+4);write('Nombre : pluriel') end;
                                                  #13 : if (existam = false)
                                                        then begin sound(220); delay(200); nosound end
                                                        else begin ok := true; gotoxy(40,ligne+4);insline;
                                                                   case noeudel.nbr2vel of
                                                                   sing : write('Nombre : singulier');
                                                                   pluriel : write('Nombre : pluriel') end
                                                             end;
                                              else       begin sound (220); delay (200); nosound end end
                                            end;
                                        gotoxy (1, ligne + 5); delline
                                  end
end;
gotoxy (1, ligne + 5);
writeln ('Genre de ce verbe ? Commun (''1''), F{minin (''2''), Masculin-F{minin (''3''),');
write ('Masculin (''4''), Masculin-Neutre (''5''), Neutre (''6''), Nul (''7'') : _');
gotoxy (wherex - 1, wherey); ok := false;
if (existam = true)
then case noeudel.genrevel of
         commun : write('1 ? '); fem : write ('2 ? '); mascfem : write ('3 ? '); masc : write ('4 ? ');
         mascneutre : write ('5 ? '); neutre : write ('6 ? '); genrenul : write ('7 ? ') end;
while (ok = false) do
if (keypressed)
then begin
      ch := readkey;
      case ch of
        '1' : begin ok := true; noeudel.genrevel := commun; gotoxy (1, ligne + 5);
                    insline; write ('Genre de ce verbe : commun') end;
        '2' : begin ok := true; noeudel.genrevel := fem; gotoxy (1, ligne + 5);
                    insline; write ('Genre de ce verbe : f{minin') end;
        '3' : begin ok := true; noeudel.genrevel := mascfem; gotoxy (1, ligne + 5);
                    insline; write ('Genre de ce verbe : masculin-f{minin') end;
        '4' : begin ok := true; noeudel.genrevel := masc; gotoxy (1,ligne + 5);
                    insline; write ('Genre de ce verbe : masculin') end;
        '5' : begin ok := true; noeudel.genrevel := mascneutre; gotoxy (1, ligne + 5);
                    insline; write ('Genre de ce verbe : masculin-neutre') end;
        '6' : begin ok := true; noeudel.genrevel := neutre; gotoxy (1, ligne + 5);
                    insline; write ('Genre de ce verbe : neutre') end;
        '7' : begin ok := true; noeudel.genrevel:= genrenul; gotoxy (1, ligne + 5);
                    insline; write ('Genre de ce verbe : -') end;
        #13 : if (existam = false)
              then begin sound(220); delay(200); nosound end
              else begin ok := true; gotoxy (1,ligne+5); insline;
                         case noeudel.genrevel of
                              commun : write('Genre de ce verbe : commun');
                              fem : write ('Genre de ce verbe : f{minin');
                              mascfem : write ('Genre de ce verbe : masculin-f{minin');
                              masc : write ('Genre de ce verbe : masculin');
                              mascneutre : write ('Genre de ce verbe : masculin-neutre');
                              neutre : write ('Genre de ce verbe : neutre');
                              genrenul : write ('Genre de ce verbe : -') end
                   end;
        else begin sound (220); delay (200); nosound end end
    end;
gotoxy (1, ligne + 6); delline; delline; ok := false;
write ('Fonction de ce verbe ? ''P''rincipale, ''S''ubordonn{e ou ''N''ulle : _'); gotoxy (wherex - 1, wherey);
if (existam = true)
```

```
      then case noeudel.fonctionvel of
              princ : write('P ? '); subord : write ('S ? '); fonctionnul : write ('N ? ') end;
      while (ok = false) do
      if (keypressed)
      then begin
            ch := readkey;
            case ch of
                 'P', 'p' : begin write (ch); ok := true; noeudel.fonctionvel := princ end;
                 'S', 's' : begin write (ch); ok := true; noeudel.fonctionvel := subord end;
                 'N', 'n' : begin write (ch); ok := true; noeudel.fonctionvel := fonctionnul end;
                     #13 : if (existam = false) then begin sound (220); delay (200); nosound end
                                                else ok := true;
                 else begin sound (220); delay (200); nosound end end
           end
end;

(*-------------------------------------------------------------------------*)

procedure AFFICH_AM_V_EL;

begin gotoxy (1, ligne); write ('Conjugaison de ce verbe : ');
      case noeudel.conjvel of
       c1 : write('1['); c2 : write('2['); c3 : write('3['); c4 : write('4['); c5 : write('4[ bis'); c6 : write('anomal') end;
      gotoxy (40, ligne); write ('Voix : ');
      case noeudel.voixvel of
      actif : write('actif'); passif : write('passif'); deponent : write('deponent'); semideponent : write('semideponent')end;
      case noeudel.modevel of
           indic, imp, subj, inf, supinum, supinu :
                      begin
                           gotoxy (1, ligne+1); write ('Personne : ');
                           case noeudel.persvel of
                               p1 : write('1['); p2 : write('2['); p3 : write('3['); pnul : write('-') end;
                           gotoxy (40, ligne+1); write ('Nombre : ');
                           case noeudel.nbr1vel of
                               sing : write('singulier'); pluriel : write('pluriel'); nombrenul : write('-') end
                      end;
           part, adjverb, gerondif :
                      begin
                           gotoxy (1, ligne+1); write ('Cas : ');
                           case noeudel.casvel of
                               nom : write ('nominatif'); voc    : write ('vocatif');     acc : write ('accusatif');
                               gen : write ('g{nitif');   datif  : write ('datif');       abl : write ('ablatif');
                               loc : write ('locatif');   indecl : write ('ind{clinable') end;
                           gotoxy (40, ligne+1); write ('Nombre : ');
                           case noeudel.nbr2vel of
                               sing : write('singulier'); pluriel : write('pluriel'); nombrenul : write('-') end
                      end
      end;
      gotoxy (1, ligne+2); write ('Mode : ');
      case noeudel.modevel of
           indic : write ('indicatif'); imp     : write ('imp{ratif');       subj     : write ('subjonctif');
           part  : write ('participe'); adjverb : write ('adjectif verbal'); gerondif : write ('g{rondif');
           inf   : write ('infinitif'); supinum : write ('supin en ''um'''); supinu   : write ('supin en ''u''') end;
      gotoxy (40, ligne+2); write ('Temps : ');
      case noeudel.tempsvel of
           pres   : write ('pr{sent');    impft       : write ('imparfait');        futspl : write ('futur simple');
           prft   : write ('parfait');    plusqueprft : write ('plus que parfait'); futant : write ('futur ant{rieur');
           fuisse : write ('''fuisse'''); tempsnul    : write ('-') end;
      gotoxy (1, ligne+3); write ('Genre : ');
      case noeudel.genrevel of
           commun   : write ('commun');   fem        : write ('f{minin');         mascfem : write ('masculin-f{minin');
           masc     : write ('masculin'); mascneutre : write ('masculin-neutre'); neutre  : write ('neutre');
           genrenul : write ('-') end;
      gotoxy (40, ligne+3); write ('Fonction : ');
      case noeudel.fonctionvel of subord : write ('subordonn{e'); princ : write ('principal'); fonctionnul : write ('-') end
```

```
end;

(*-------------------------------------------------------------------------*)

procedure SELECTAM;

var selection, ok, oke, existam : boolean;
    newlnael : listnoeudael;
    ch, ch2 : char;
    categoriemot : typecategorie;

label FIN;

begin
     remonte := false;
     existam := false;
     selection := false;
     while (selection = false) do
     begin
       clrscr; gotoxy (1, 2); delline; write (entete, motanalyse);
       if (existfnael = false)
       then begin
              gotoxy (1, 3);
              writeln ('Tape ''F5'' pour remonter ou r{ponds @ la question suivante : ');
              writeln ('Quelle est la cat{gorie grammaticale de ce mot ', motanalyse, ' ?');
              writeln ('Substantif (''1''), Adjectif (''2''), Adjectif pronom (''3''), Pr{position (''4'')');
              write ('ou Conjonction (''5'') : _');
              gotoxy (wherex - 1, wherey); ok := false;
              if (existam = true)
              then case noeudfilscourel^.noeudel.categorieel of
                       subst : write ('1 ? '); adj : write ('2 ? '); adjpron : write ('3 ? ');
                       prep : write ('4 ? '); conj : write ('5 ? ') end;
              while (ok = false) do
              begin
                repeat until keypressed;
                ch := readkey;
                case ch of
                     '1' : begin ok := true; gotoxy (1,3); insline; write ('Cat{gorie : substantif');
                                 categoriemot := subst; if (existam = true)
                                                        then if (noeudfilscourel^.noeudel.categorieel <> categoriemot)
                                                             then existam := false end;
                     '2' : begin ok := true; gotoxy (1,3); insline; write ('Cat{gorie : adjectif');
                                 categoriemot := adj; if (existam = true)
                                                      then if (noeudfilscourel^.noeudel.categorieel <> categoriemot)
                                                           then existam := false end;
                     '3' : begin ok := true;gotoxy (1,3); insline;write ('Cat{gorie : adjectif pronom');
                                 categoriemot := adjpron; if (existam = true)
                                                          then if (noeudfilscourel^.noeudel.categorieel <> categoriemot)
                                                               then existam := false end;
                     '4' : begin ok := true; gotoxy (1,3); insline; write ('Cat{gorie : pr{position');
                                 categoriemot := prep; if (existam = true)
                                                       then if (noeudfilscourel^.noeudel.categorieel <> categoriemot)
                                                            then existam := false end;
                     '5' : begin ok := true; gotoxy (1,3); insline; write ('Cat{gorie : conjonction');
                                 categoriemot := conj; if (existam = true)
                                                       then if (noeudfilscourel^.noeudel.categorieel <> categoriemot)
                                                            then existam := false end;
                     #0  : if keypressed then begin
                                              ch2 := readkey;
                                              if (ch2 = #63) then begin ok := true; remonte := true; goto FIN end
                                                             else begin sound (220); delay (200); nosound end
                                             end
                                           else begin sound (220); delay (200); nosound end;
                     #13 : if (existam = false)
                           then begin sound(220); delay(200); nosound end
```

```
                    else begin ok := true; gotoxy (1,3); insline;
                              case noeudfilscourel^.noeudel.categorieel of
                                   subst : write ('Cat{gorie : substantif');
                                   adj : write ('Cat{gorie : adjectif');
                                   adjpron : write ('Cat{gorie : adjectif pronom');
                                   prep : write ('Cat{gorie : pr{position');
                                   conj : write ('Cat{gorie : conjonction') end
                         end;
               else  begin sound (220); delay (200); nosound end
          end
        end;
        gotoxy (1, 4) ; delline; delline; delline; delline;
        selection := true;
        if (existam = false)
        then begin
              new (newlnael); newlnael^.noeudfilsel := nil;
              newlnael^.noeudpereel := lnacourel; newlnael^.noeudel.typenoeudel := n2;
              newlnael^.noeudel.lemmeel := ''; newlnael^.noeudel.categorieel := categoriemot;
              case categoriemot of
                   subst   : REMPLI_AM_SUBST_EL (newlnael^.noeudel, 4, existam);
                   adj     : REMPLI_AM_ADJ_EL (newlnael^.noeudel, 4, existam);
                   adjpron : REMPLI_AM_ADJPRON_EL (newlnael^.noeudel, 4, existam);
                   prep    : REMPLI_AM_PREP_EL (newlnael^.noeudel, 4, existam);
                   conj : REMPLI_AM_CONJ_EL (newlnael^.noeudel, 4, existam) end;
              lnacourel^.noeudfilsel := newlnael;
              noeudfilscourel := newlnael
             end
        else case noeudfilscourel^.noeudel.categorieel of
                  subst   : REMPLI_AM_SUBST_EL (noeudfilscourel^.noeudel, 4, existam);
                  adj     : REMPLI_AM_ADJ_EL (noeudfilscourel^.noeudel, 4, existam);
                  adjpron : REMPLI_AM_ADJPRON_EL (noeudfilscourel^.noeudel, 4, existam);
                  prep    : REMPLI_AM_PREP_EL (noeudfilscourel^.noeudel, 4, existam);
                  conj : REMPLI_AM_CONJ_EL (noeudfilscourel^.noeudel, 4, existam) end
    end
else begin
          categoriemot := noeudfilscourel^.noeudel.categorieel;
          case categoriemot of
           subst : AFFICH_AM_SUBST_EL(noeudfilscourel^.noeudel,4); adj : AFFICH_AM_ADJ_EL(noeudfilscourel^.noeudel,4);
           adjpron : AFFICH_AM_ADJPRON_EL(noeudfilscourel^.noeudel,4);
           prep : AFFICH_AM_PREP_EL(noeudfilscourel^.noeudel,4); conj : AFFICH_AM_CONJ_EL(noeudfilscourel^.noeudel,4)
          end;
          writeln; writeln; write ('S{lection ? ''O''-''N'' ou Remont{e ''F5'' : _');
          gotoxy (wherex - 1, wherey); ok := false;
          while (ok = false) do
          begin
            if keypressed
            then begin
                  ch := readkey;
                  case ch of
                       'O', 'o' : begin ok := true; selection := true; write (ch) end;
                       'N', 'n' : begin
                                    ok := true; write (ch);
                                    existfnael := false;
                                    existam := true
                                  end;
                           #0  : if keypressed
                                 then begin
                                        ch2 := readkey;
                                        if (ch2 = #63) then begin ok := true; remonte := true; goto FIN end
                                                       else begin sound (220); delay (200); nosound end
                                      end
                                 else begin sound (220); delay (200); nosound end;
                      else       begin sound (220); delay (200); nosound end end
                 end
          end
```

```
            end
    end;
FIN :
    write ('')
end;
```

```
Unit TRTS;

Interface
uses ('$U b:globalel ') GLOBALEL, CRT, DOS,
     ('$U b:cpltel ') CPLTEL, ('$U b:le_unit ') LE_UNIT;

type str10 = string[10];

procedure TRT_VP ( var existfnael : boolean; var firstmcp : lcontphr;
                   var lnacour : listnoeudaa; var firstlna : listnoeudaa;
                   var lnacourel : listnoeudael; var firstlnael : listnoeudael;
                   var finanbool : boolean; remonte : boolean;
                   var namefct : str14; var titretxt : str60;
                   var firstld : list_descr; var ldcour : list_descr);

procedure TRT_GN (var existfnael : boolean; var firstmcp : lcontphr; var lnacour : listnoeudaa;
                  var lnacourel : listnoeudael; var finanbool : boolean; casgn : cas;
                  remonte : boolean; var trtsuivant : traitement; var namefct : str14;
                  var titretxt : str60; var firstld : list_descr; var ldcour : list_descr);

procedure TRSF (var noeud : noeudaa; var analyse : str10);

procedure TRT_TRAD (var firstmcp : lcontphr; var accesfamml : boolean;
                    var lnacour : listnoeudaa; var finanbool : boolean);


Implementation


procedure TRT_VP;   (* verbe = verbe simple <> "esse" ou "esse" + participe pass( *)

var gotofin, gotoanal, gotodebut, remontee, ok, selection, existam : boolean;
    noeudfilscour : listnoeudaa;
    noeudfilscourel, lnaelpere, newlnael : listnoeudael;
    entete : str30;
    comment : str80;
    ch, ch2 : char;

label DEBUT, DEBUTPP, ANAL, AFTERAMV, ANALPP, AFTERAMPP, PARTICIPE, FINTRTVP;

begin
     finanbool := false;
     if (remonte = true) (* lnacourel^.noeudel = mot et noeudfilscourel^.noeudel = A.M. *)
     then begin existfnael := true; if (lnacourel^.noeudpereel = nil) then goto ANAL else goto ANALPP end;

     if (existfnael = true) then noeudfilscourel := firstlnael else begin noeudfilscourel := nil end;
DEBUT :
     entete := 'Verbe principal : '; lnaelpere := nil;
     SELECTMOT (firstmcp, existfnael, verbe, entete, noeudfilscourel, firstlnael, lnaelpere, remontee);
     if (remontee = true) then if ( (noeudfilscourel = nil) or (noeudfilscourel = firstlnael) )
                               then begin finanbool := true; goto FINTRTVP end
                               else begin noeudfilscourel := firstlnael; existfnael := true; goto DEBUT end;

(* comparaison avec l'analyse du professeur *)
     lnacourel := firstlnael;
     comment := 'R{fl{chis et trouve un verbe repr{sentant l''action principale de la phrase !';
     COMPARMOT (firstlna, lnacour, noeudfilscourel, finanbool, gotofin, firstmcp,
                lnacourel, existfnael, gotodebut, namefct, titretxt,
                firstld, ldcour, comment);
     if (gotofin = true) then goto FINTRTVP;
     if (gotodebut = true) then goto DEBUT;

     lnacourel := noeudfilscourel;
     if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
     noeudfilscourel := lnacourel^.noeudfilsel;
```

```
ANAL :      (* A.M. verbe principal *)
     entete := 'Analyse du verbe principal : ';
     remontee := false; existam := false; selection := false;
     while (selection = false) do
     begin
       clrscr; gotoxy (1, 2); delline; write (entete, lnacourel^.noeudel.motiel, '''F5'' pour remonter ou ''RETURN'''');
       ok := false;
       while (ok = false) do
       begin
         repeat until keypressed;
         ch := readkey;
         case ch of
              #13 : ok := true;
               #0 : if keypressed then begin
                                         ch2 := readkey;
                                         if (ch2 = #63) then begin ok := true; remontee := true; goto AFTERAMV end
                                                        else begin sound (220); delay (200); nosound end
                                       end
                                  else begin sound (220); delay (200); nosound end;
              else  begin sound (220); delay (200); nosound end
         end
       end;
       if (existfnael = false)
       then begin
              gotoxy (1, 4); selection := true;
              if (existam = false)
              then begin
                     new (newlnael); newlnael^.noeudfilsel := nil; newlnael^.noeudpereel := lnacourel;
                     newlnael^.noeudel.typenoeudel := n2; newlnael^.noeudel.lemmeel := ''; newlnael^.noeudel.categorieel := v;
                     REMPLI_AM_V_EL (newlnael^.noeudel, 4, existam);
                     lnacourel^.noeudfilsel := newlnael; noeudfilscourel := newlnael
                   end
              else REMPLI_AM_V_EL (noeudfilscourel^.noeudel, 4, existam)
            end
       else begin
              AFFICH_AM_V_EL (noeudfilscourel^.noeudel, 4);
              writeln; writeln; write ('S{lection ? ''O''-''N'' ou Remont{e ''F5'' : _');
              gotoxy (wherex - 1, wherey); ok := false;
              while (ok = false) do
              begin
                repeat until keypressed;
                ch := readkey;
                case ch of
                     'O', 'o' : begin ok := true; selection := true; write (ch) end;
                     'N', 'n' : begin ok := true; write (ch); existfnael := false; existam := true end;
                           #0 : if keypressed
                                then begin
                                       ch2 := readkey;
                                       if (ch2 = #63) then begin ok := true; remontee := true; goto AFTERAMV end
                                                      else begin sound (220); delay (200); nosound end
                                     end
                                else begin sound (220); delay (200); nosound end;
                          else  begin sound (220); delay (200); nosound end
                end
              end
            end
     end;
AFTERAMV :
     if (remontee = true) then begin
                                 MAJLIBRE (firstmcp, lnacourel^.noeudel, true);
                                 noeudfilscourel := lnacourel;
                                 existfnael := true; goto DEBUT
                               end;

(* comparaison avec l'analyse du professeur *)
```

```
    COMPARAM (noeudfilscour, lnacour, noeudfilscourel, lnacourel, v, finanbool,
             gotofin, gotoanal, existfnael, namefct, titretxt, firstld, ldcour);
    if (gotofin = true) then goto FINTRTVP;
    if (gotoanal = true) then goto ANAL;

PARTICIPE :
    if (lnacour^.noeud.lemme = 'SVM                ')
    then begin
          if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
          noeudfilscourel := lnacourel^.noeudfilsel;
DEBUTPP :
          entete := 'Participe Passé : ';
          SELECTMOT (firstmcp, existfnael, verbe, entete, noeudfilscourel, lnacourel^.noeudfilsel, lnacourel, remontee);
          if (remontee = true) then begin
                                     noeudfilscourel := lnacourel; lnacourel := lnacourel^.noeudpereel;
                                     noeudfilscour := lnacour; lnacour := lnacour^.noeudpere;
                                     existfnael := true; goto ANAL
                                    end;

          (* comparaison avec l'analyse du professeur *)
          comment := '';
          COMPARMOT (lnacour^.noeudfils, noeudfilscour, noeudfilscourel, finanbool,
                    gotofin, firstmcp, lnacourel^.noeudfilsel, existfnael,
                    gotodebut, namefct, titretxt, firstld, ldcour, comment);
          if (gotofin = true) then goto FINTRTVP;
          if (gotodebut = true) then goto DEBUTPP;

          lnacourel := noeudfilscourel; lnacour := noeudfilscour;
          if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
          noeudfilscourel := lnacourel^.noeudfilsel;
ANALPP :  (* A.M. participe passé *)
          entete := 'Analyse du participe passé ';
          remontee := false; existam := false; selection := false;
          while (selection = false) do
          begin
          clrscr; gotoxy (1, 2); delline; write (entete, lnacourel^.noeudel.motlel, '''F5'' pour remonter ou ''RETURN''');
          ok := false;
          while (ok = false) do
          begin
            repeat until keypressed;
            ch := readkey;
            case ch of
                 #13 : ok := true;
                  #0 : if keypressed then begin
                                            ch2 := readkey;
                                            if (ch2 = #63) then begin ok := true; remontee := true; goto AFTERAMV end
                                                           else begin sound (220); delay (200); nosound end
                                          end
                                     else begin sound (220); delay (200); nosound end;
                 else  begin sound (220); delay (200); nosound end
           end
          end;
          if (existfnael = false)
          then begin
                 gotoxy (1, 4); selection := true;
                 if (existam = false)
                 then begin
                        new (newlnael); newlnael^.noeudfilsel := nil; newlnael^.noeudpereel := lnacourel;
                        newlnael^.noeudel.typenoeudel := n2; newlnael^.noeudel.lemmeel := '';
                        newlnael^.noeudel.categorieel := v;
                        REMPLI_AM_V_EL (newlnael^.noeudel, 4, existam);
                        lnacourel^.noeudfilsel := newlnael; noeudfilscourel := newlnael
                      end
                 else REMPLI_AM_V_EL (noeudfilscourel^.noeudel, 4, existam)
               end
```

```
          else begin
                AFFICH_AM_V_EL (noeudfilscourel^.noeudel, 4);
                writeln; writeln; write ('S{lection ? ''O''-''N'' ou Remont{e ''F5'' : _');
                gotoxy (wherex - 1, wherey); ok := false;
                while (ok = false) do
                begin
                  repeat until keypressed;
                  ch := readkey;
                  case ch of
                       'O', 'o' : begin ok := true; selection := true; write (ch) end;
                       'N', 'n' : begin ok := true; write (ch); existfnael := false; existam := true end;
                            #0 : if keypressed
                                 then begin
                                        ch2 := readkey;
                                        if (ch2 = #63) then begin ok := true; remontee := true; goto AFTERAMV end
                                                       else begin sound (220); delay (200); nosound end
                                      end
                                 else begin sound (220); delay (200); nosound end;
                          else  begin sound (220); delay (200); nosound end
                  end
                end
              end
          end;
AFTERAMPP :
          if (remontee = true) then begin
                                      MAJLIBRE (firstmcp, lnacourel^.noeudel, true);
                                      noeudfilscourel := lnacourel; lnacourel := lnacourel^.noeudpereel;
                                      noeudfilscour := lnacour; lnacour := lnacour^.noeudpere;
                                      existfnael := true; goto DEBUTPP
                                    end;

          (* comparaison avec l'analyse du professeur *)
          COMPARAM (noeudfilscour, lnacour, noeudfilscourel, lnacourel,
                    v, finanbool, gotofin, gotoanal, existfnael, namefct,
                    titretxt, firstld, ldcour);
          if (gotofin = true) then goto FINTRTVP;
          if (gotoanal = true) then goto ANALPP
        end;
FINTRTVP :
    write ('')
end;

(*-------------------------------------------------------------------------*)

procedure TRT_GN;

(* traitement GN : choix 'centre-gn', analyse morphologique et d{coupe     *)
(* syntaxique suivie de la s{lection et de l'analyse de chaque mot.        *)

var ok, oke, trouve, gotoanal, gotofin, gotodebut, selection, right, remontee, libre : boolean;
    mcpcour, ln3, ln3cour, firstln3 : lcontphr;
    noeudfilscour, lnatemp : listnoeudaa;
    k, i : integer;
    mememot : str20;
    ch, ch2 : char;
    fonctionmot : fonctiontype;
    noeudfilscourel, newlnael : listnoeudael;
    entete : str30;
    comment : str80;

label  FINTRTGN, DEBUT, ANAL, CHOIXGN, ANALGN;

begin
    finanbool := false; trtsuivant := trtnul;
    if (remonte = true) then begin noeudfilscourel := lnacourel^.noeudfilsel; goto ANALGN end;
```

```
    if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
    noeudfilscourel := lnacourel^.noeudfilsel;
DEBUT :
    case casgn of
        nom : begin entete := 'Centre GNsujet : ';
                SELECTMOT(firstmcp,existfnael,sujet,entete,noeudfilscourel,lnacourel^.noeudfilsel,lnacourel,remontee) end;
        acc : begin entete := 'Centre GNcod : ';
                SELECTMOT(firstmcp,existfnael,cod,entete,noeudfilscourel,lnacourel^.noeudfilsel,lnacourel,remontee) end;
        abl : begin entete := 'Centre GNcompl{ment : ';
                SELECTMOT(firstmcp,existfnael,cplt,entete,noeudfilscourel,lnacourel^.noeudfilsel,lnacourel,remontee) end
    end;
    if (remontee = true) then begin
                            libre := true;
                            noeudfilscourel := lnacourel; lnacourel := lnacourel^.noeudpereel;
                            noeudfilscour := lnacour; lnacour := lnacour^.noeudpere;
                            case casgn of
                                nom : trtsuivant := trtv; acc : trtsuivant := trtgns; abl : trtsuivant := trtgncod end;
                            existfnael := true; goto FINTRTGN
                          end
                     else libre := false;

    comment := 'R{fl{chis et trouve qui agit sur le verbe !';
    COMPARMOT (lnacour^.noeudfils, noeudfilscour, noeudfilscourel, finanbool,
              gotofin, firstmcp, lnacourel^.noeudfilsel, existfnael, gotodebut,
              namefct, titretxt, firstld, ldcour, comment);
    if (gotofin = true) then goto FINTRTGN;
    if (gotodebut = true) then begin libre := true; goto DEBUT end;

    lnacourel := noeudfilscourel; lnacour := noeudfilscour;
    if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
    noeudfilscourel := lnacourel^.noeudfilsel;
ANAL :
    case casgn of
        nom : entete := 'Analyse du centre du GNsujet ';
        acc : entete := 'Analyse du centre du GNcod ';
        abl : entete := 'Analyse du centre du GNcplt '
    end;
    SELECTAM (entete,lnacourel^.noeudel.motiel,existfnael,noeudfilscourel,lnacourel, remontee);
    if (remontee = true) then begin
                            if (libre = false)
                            then begin MAJLIBRE (firstmcp, lnacourel^.noeudel, true); libre := true end;
                            noeudfilscourel := lnacourel; lnacourel := lnacourel^.noeudpereel;
                            noeudfilscour := lnacour; lnacour := lnacour^.noeudpere;
                            existfnael := true; goto DEBUT
                           end;

    COMPARAM (noeudfilscour, lnacour, noeudfilscourel, lnacourel, noeudfilscourel^.noeudel.categorieel, finanbool,
             gotofin, gotoanal, existfnael, namefct, titretxt, firstld, ldcour);
    if (gotofin = true) then goto FINTRTGN;
    if (gotoanal = true) then goto ANAL;

    if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
    if (libre = false) then begin MAJLIBRE (firstmcp, lnacourel^.noeudpereel^.noeudel, true); libre := true end;
                                (* relib{rer {l{ment central *)
    noeudfilscourel := lnacourel^.noeudfilsel;
CHOIXGN :
    selection := false;
    while (selection = false) do
    begin
         clrscr;
         if (existfnael = true)
         then begin
               AFFICHN3 (noeudfilscourel^.noeudel, firstmcp, 1, 8);
               gotoxy (1, 10); delline;
```

```
            case casgn of
                nom : write ('S{lection de ce GNsujet ? ''O''-''N'' ou Remont{e ''F5'' : _');
                acc : write ('S{lection de ce GNcod ? ''O''-''N'' ou Remont{e ''F5'' : _');
                abl : write ('S{lection de ce GNcompl{ment ? ''O''-''N'' ou Remont{e ''F5'' : _');
            end;
            gotoxy (wherex - 1, wherey); ok := false;
            while (ok = false) do
            begin
              if keypressed then begin
                                ch := readkey;
                                case ch of
                                    'O', 'o' : begin ok := true; selection := true; write (ch) end;
                                    'N', 'n' : begin ok := true; write (ch); existfnael := false end;
                                    #0       : if keypressed
                                               then begin
                                                      ch2 := readkey;
                                                      if (ch2 = #63)
                                                      then begin
                                                             ok := true;
                                                             noeudfilscourel := lnacourel;
                                                             lnacourel := lnacourel^.noeudpereel;
                                                             noeudfilscour := lnacour;
                                                             lnacour := lnacour^.noeudpere;
                                                             existfnael := true; goto ANAL
                                                           end
                                                      else begin sound (220); delay (200); nosound end
                                                    end
                                               else begin sound (220); delay (200); nosound end;
                                    else       begin sound (220); delay (200); nosound end
                                end
                               end
        end
      end
else begin
        gotoxy (1, 2);
        case casgn of
            nom : write ('Tape ''RETURN'' pour choisir le GNsujet ou ''F5'' pour remonter ');
            acc : write ('Tape ''RETURN'' pour choisir le GNcod ou ''F5'' pour remonter ');
            abl : write ('Tape ''RETURN'' pour choisir le GNcompl{ment ou ''F5'' pour remonter ')
        end;
        ok := false;
        while (ok = false) do
        begin
          repeat until keypressed;
          ch := readkey;
          case ch of
              #13 : ok := true;
              #0  : if keypressed
                    then begin
                           ch2 := readkey;
                           if (ch2 = #63)
                           then begin
                                  ok := true;
                                  noeudfilscourel := lnacourel;
                                  lnacourel := lnacourel^.noeudpereel;
                                  noeudfilscour := lnacour;
                                  lnacour := lnacour^.noeudpere;
                                  existfnael := true; goto ANAL
                                end
                           else begin sound (220); delay (200); nosound end
                         end
                    else begin sound (220); delay (200); nosound end;
              else  begin sound (220); delay (200); nosound end
          end
        end;
```

```
                selection := true;
                new (newlnael); newlnael^.noeudfilsel := nil;
                newlnael^.noeudpereel := lnacourel; newlnael^.noeudel.typenoeudel := n3;
                case casgn of
                     nom : newlnael^.noeudel.symbcatsyntel := gnsujet;
                     acc : newlnael^.noeudel.symbcatsyntel := gncod;
                     abl : newlnael^.noeudel.symbcatsyntel := gnabl
                end;
                ln3cour := nil; mcpcour := firstmcp;
                while (mcpcour <> nil) do
                begin
                  if (mcpcour^.libre = true)
                  then begin
                         gotoxy (1, 2); delline;
                         case casgn of
                              nom : write ('Mot appartenant au GNsujet : ', mcpcour^.mot, ' ? ''O''-''N'' : _');
                              acc : write ('Mot appartenant au GNcod : ', mcpcour^.mot, ' ? ''O''-''N'' : _');
                              abl : write ('Mot appartenant au GNcompl{ment : ', mcpcour^.mot, ' ? ''O''-''N'' : _')
                         end;
                         gotoxy (wherex - 1, wherey); oke := false;
                         while (oke = false) do
                            if keypressed then begin
                                               ch := readkey;
                                               case ch of
                                                    'O', 'o' : begin
                                                                 oke := true; write (ch);
                                                                 new (ln3); ln3^.mot := mcpcour^.mot;
                                                                 ln3^.libre := true; ln3^.suivant := nil;
                                                                 if (ln3cour = nil)
                                                                 then begin firstln3 := ln3; ln3cour := ln3 end
                                                                 else begin ln3cour^.suivant := ln3;
                                                                            ln3cour := ln3cour^.suivant
                                                                      end
                                                               end;
                                                    'N', 'n' : begin oke := true; write (ch) end;
                                                    else '     begin sound (220); delay (200); nosound end
                                               end
                                             end;
                       end;
                  mcpcour := mcpcour^.suivant
                end;
                AFFICHLN3 (firstln3, 1, 8);
                REMPLI_N3_LN3 (firstln3, newlnael^.noeudel.ensmots3el, firstmcp);
                lnacourel^.noeudfilsel := newlnael; noeudfilscourel := newlnael
              end;
end;

(* comparaison avec analyse du professeur *)
noeudfilscour := lnacour^.noeudfils; trouve := false;
while ( (noeudfilscour <> nil) and (trouve = false) ) do
begin
  ok := true; i := 1;
  while ( (i < 85) and (ok = true) ) do
     if (noeudfilscour^.noeud.ensmots3[i] = noeudfilscourel^.noeudel.ensmots3el[i])
     then i := i + 1 else ok := false;
  if (ok = true) then trouve := true else noeudfilscour := noeudfilscour^.noeudfrere
end;
if (trouve = false)
then begin
       gotoxy (1, 10); delline;
       writeln ('Erreur ! Tape une touche pour faire une autre proposition ou ESC !');
       case casgn of
            nom : write ('R{fl{chis et trouve l''ensemble des mots agissant sur le verbe !');
            acc : write ('R{fl{chis et trouve l''ensemble des mots ''COD'' du verbe !');
            abl : write ('R{fl{chis et trouve l''ensemble des mots ''compl{ment'' du verbe !');
```

```
        end;
        repeat until keypressed;
        ch := readkey;
        if (ch = #27) then begin finanbool := true; goto FINTRTGN end;
        noeudfilscourel := lnacourel^.noeudfilsel;
        existfnael := true; goto CHOIXGN
      end
  else begin
        if (noeudfilscour^.noeud.accept_comment = true)
        then begin
              gotoxy (1, 10); delline;
              if (noeudfilscour^.noeud.comment_specif <> '')
              then write (noeudfilscour^.noeud.comment_specif)
              else write (noeudfilscour^.noeud.comment_syst)
             end;
        if (noeudfilscour^.noeud.ind_brk = break)
        then begin
              gotoxy (1, 12);
              write ('ERREUR IMPARDONNABLE ! Tape une touche pour corriger ton analyse ou ESC !');
              repeat until keypressed; ch := readkey;
              if (ch = #27) then begin finanbool := true; goto FINTRTGN end;
              noeudfilscourel := lnacourel^.noeudfilsel;
              existfnael := true; goto CHOIXGN
             end
        else begin
              ok := false;
              while (ok = false) do
              begin
                gotoxy (1, 13); delline; delline; delline;
                writeln ('Tape F1 pour voir le texte, F2 pour voir le commentaire du texte, F3 pour voir');
                writeln ('le commentaire de la phrase, F4 pour modifier ton choix, ESC pour partir');
                write ('ou une autre touche pour continuer !');
                repeat until keypressed; ch := readkey;
                if (ch = #27)
                then begin finanbool := true; ok := true; goto FINTRTGN end
                else if ( (ch = #0) and keypressed)
                     then begin
                           ch2 := readkey;
                           case ch2 of
                                #59 : VISUTXTPHR (namefct, 0, titretxt, 1, 13, right);
                                #60 : begin
                                        gotoxy (1, 13); delline; delline; delline;
                                        write (firstld^.contenu.comment_txt); gotoxy (40, 15);
                                        write ('Tape une touche pour continuer !');
                                        repeat until keypressed; ch := readkey
                                      end;
                                #61 : begin
                                        gotoxy (1, 13); delline; delline; delline;
                                        write (ldcour^.contenu.comment_phrase); gotoxy (40, 15);
                                        write ('Tape une touche pour continuer !');
                                        repeat until keypressed; ch := readkey
                                      end;
                                #62 : begin
                                        ok := true; noeudfilscourel := lnacourel^.noeudfilsel;
                                        existfnael := true; goto CHOIXGN
                                      end;
                                else  begin sound (220); delay (200); nosound end
                           end
                          end
                     else ok := true
              end;

              (* MAJLIBRE de tous les {l{ments du GN *)
              for i := 1 to 85 do
              begin
```

```
        if (noeudfilscourel^.noeudel.ensmots3el[i] = true)
        then begin
                mcpcour := firstmcp; k := 1;
                while (k < i) do begin mcpcour := mcpcour^.suivant; k := k + 1 end;
                if (mcpcour^.libre = true)
                then mcpcour^.libre := false
                else begin
                        mememot := mcpcour^.mot;
                        repeat mcpcour := mcpcour^.suivant
                        until ( (mcpcour^.mot = mememot) and (mcpcour^.libre = true) );
                        mcpcour^.libre := false
                     end
             end
    end;
    libre := false;
    lnacourel := noeudfilscourel; lnacour := noeudfilscour;

    (* analyse morphologique de tous les {l{ments du GN *)
    (* pas besoin de travailler avec une listeln3, il suffit de suivre lnacour *)
    case casgn of nom : fonctionmot := sujet; acc : fonctionmot := cod; abl : fonctionmot := cplt end;
    while (lnacour^.noeudfils^.noeud.fonction = fonctionmot) do
    begin
      if (lnacourel^.noeudfilsel <> nil) then existfnael := true else existfnael := false;
      lnacour := lnacour^.noeudfils; noeudfilscourel := lnacourel^.noeudfilsel;
ANALGN :
      case casgn of
           nom : entete := 'Analyse du mot du GNsujet ';
           acc : entete := 'Analyse du mot du GNcod ';
           abl : entete := 'Analyse du mot du GNcplt '
      end;
      SELECTAM (entete,lnacour^.noeud.mot1,existfnael,noeudfilscourel,lnacourel,remontee);
      if (remontee = true) then begin
                                  noeudfilscourel := lnacourel; lnacourel := lnacourel^.noeudpereel;
                                  lnacour := lnacour^.noeudpere^.noeudpere; existfnael := true;
                                  if (noeudfilscourel^.noeudel.typenoeudel = n3)
                                  then begin
                                         libre := true;
                                         for i := 1 to 85 do
                                         begin
                                           if (noeudfilscourel^.noeudel.ensmots3el[i] = true)
                                           then begin
                                                  mcpcour := firstmcp; k := 1;
                                                  while (k < i) do
                                                  begin mcpcour := mcpcour^.suivant; k := k + 1 end;
                                                  if (mcpcour^.libre = false)
                                                  then mcpcour^.libre := true
                                                  else begin
                                                         mememot := mcpcour^.mot;
                                                         repeat mcpcour := mcpcour^.suivant
                                                         until ( (mcpcour^.mot = mememot)
                                                         and (mcpcour^.libre = false) );
                                                         mcpcour^.libre := true
                                                       end
                                                end
                                         end;
                                         goto CHOIXGN
                                       end
                                  else goto ANALGN
                                end;

      COMPARAM (noeudfilscour, lnacour, noeudfilscourel, lnacourel, noeudfilscourel^.noeudel.categorieel,
               finanbool, gotofin, gotoanal, existfnael, namefct, titretxt, firstld, ldcour);
      if (gotofin = true) then goto FINTRTGN;
      if (gotoanal = true) then goto ANALGN
    end
```

```
                end
        end;
FINTRTGN :
    write ('')
end;

{'-------------------------------------------------------------------------'}

procedure TRSF;

begin
     analyse := '          ';
     case noeud.categorie of
          subst : begin
                       analyse[1] := '1';
                       case noeud.declsubst of
                            d1 : analyse[2] := '1'; d2 : analyse[2] := '2'; d3 : analyse[2] := '3';
                            d4 : analyse[2] := '4'; d5 : analyse[2] := '5'; d6 : analyse[2] := '6';
                            d7 : analyse[2] := '7' end;
                       case noeud.cassubst of
                            nom    : case noeud.nbrsubst of sing : analyse[3] := 'A'; pluriel : analyse[3] := 'J' end;
                            voc    : case noeud.nbrsubst of sing : analyse[3] := 'B'; pluriel : analyse[3] := 'K' end;
                            acc    : case noeud.nbrsubst of sing : analyse[3] := 'C'; pluriel : analyse[3] := 'L' end;
                            gen    : case noeud.nbrsubst of sing : analyse[3] := 'D'; pluriel : analyse[3] := 'M' end;
                            datif  : case noeud.nbrsubst of sing : analyse[3] := 'E'; pluriel : analyse[3] := 'N' end;
                            abl    : case noeud.nbrsubst of sing : analyse[3] := 'F'; pluriel : analyse[3] := 'O' end;
                            loc    : case noeud.nbrsubst of sing : analyse[3] := 'G'; pluriel : analyse[3] := 'P' end;
                            indecl : analyse[3] := 'Z' end;
                       analyse[4] := '0'; analyse[5] := '0';
                       analyse[6] := ' '; analyse[7] := ' ';
                       case noeud.genresubst of
                            commun   : analyse[8] := '1'; fem        : analyse[8] := '2'; mascfem : analyse[8] := '3';
                            masc     : analyse[8] := '4'; mascneutre : analyse[8] := '5'; neutre  : analyse[8] := '6';
                            genrenul : analyse[8] := ' ' end;
                       analyse[9] := ' '; analyse[10] := ' ';
                end;
          adj : begin
                       analyse[1] := '2';
                       case noeud.classeadj of
                         a1 : case noeud.degreadj of
                                positif : analyse[2]:='1'; comparatif : analyse[2]:='A'; superlatif : analyse[2]:='J' end;
                         a2 : case noeud.degreadj of
                                positif : analyse[2]:='2'; comparatif : analyse[2]:='B'; superlatif : analyse[2]:='K' end;
                         a3 : case noeud.degreadj of
                                positif : analyse[2]:='3'; comparatif : analyse[2]:='C'; superlatif : analyse[2]:='L' end;
                         a4 : case noeud.degreadj of
                                positif : analyse[2]:='4'; comparatif : analyse[2]:='D'; superlatif : analyse[2]:='M' end;
                         a5 : case noeud.degreadj of
                                positif : analyse[2]:='5'; comparatif : analyse[2]:='E'; superlatif : analyse[2]:='N' end;
                         a6 : case noeud.degreadj of
                                positif : analyse[2]:='6'; comparatif : analyse[2]:='F'; superlatif : analyse[2]:='O' end;
                         a7 : analyse[2] := '7';
                       end;
                       case noeud.casadj of
                            nom    : case noeud.nbradj of sing : analyse[3] := 'A'; pluriel : analyse[3] := 'J' end;
                            voc    : case noeud.nbradj of sing : analyse[3] := 'B'; pluriel : analyse[3] := 'K' end;
                            acc    : case noeud.nbradj of sing : analyse[3] := 'C'; pluriel : analyse[3] := 'L' end;
                            gen    : case noeud.nbradj of sing : analyse[3] := 'D'; pluriel : analyse[3] := 'M' end;
                            datif  : case noeud.nbradj of sing : analyse[3] := 'E'; pluriel : analyse[3] := 'N' end;
                            abl    : case noeud.nbradj of sing : analyse[3] := 'F'; pluriel : analyse[3] := 'O' end;
                            loc    : case noeud.nbradj of sing : analyse[3] := 'G'; pluriel : analyse[3] := 'P' end;
                            indecl : analyse[3] := 'Z' end;
                       analyse[4] := '0'; analyse[5] := '0';
                       analyse[6] := ' '; analyse[7] := ' ';
                       case noeud.genreadj of
```

```
                    commun   : analyse[8] := '1'; fem       : analyse[8] := '2'; mascfem : analyse[8] := '3';
                    masc     : analyse[8] := '4'; mascneutre : analyse[8] := '5'; neutre  : analyse[8] := '6';
                    genrenul : analyse[8] := ' ' end;
                analyse[9] := ' '; analyse[10] := ' ';
        end;
num : begin
                analyse[1] := '3';
                case noeud.catnum of
                    card : analyse[2] := '1';
                    ord : case noeud.degrenum of
                              positif    : analyse[2] := '2'; comparatif : analyse[2] := 'B';
                              superlatif : analyse[2] := 'K'  end;
                    distr : analyse[2] := '3';
                    mult : analyse[2] := '4';
                    advord : case noeud.degrenum of
                                 positif    : analyse[2] := '5'; comparatif : analyse[2] := 'E';
                                 superlatif : analyse[2] := 'N'  end;
                    advmult : analyse[2] := '6'
                end;
                case noeud.casnum of
                    nom    : case noeud.nbrnum of sing : analyse[3] := 'A'; pluriel : analyse[3] := 'J' end;
                    voc    : case noeud.nbrnum of sing : analyse[3] := 'B'; pluriel : analyse[3] := 'K' end;
                    acc    : case noeud.nbrnum of sing : analyse[3] := 'C'; pluriel : analyse[3] := 'L' end;
                    gen    : case noeud.nbrnum of sing : analyse[3] := 'D'; pluriel : analyse[3] := 'M' end;
                    datif  : case noeud.nbrnum of sing : analyse[3] := 'E'; pluriel : analyse[3] := 'N' end;
                    abl    : case noeud.nbrnum of sing : analyse[3] := 'F'; pluriel : analyse[3] := 'O' end;
                    loc    : case noeud.nbrnum of sing : analyse[3] := 'G'; pluriel : analyse[3] := 'P' end;
                    indecl : analyse[3] := 'Z' end;
                analyse[4] := '0'; analyse[5] := '0';
                analyse[6] := ' '; analyse[7] := ' ';
                case noeud.genrenum of
                    commun   : analyse[8] := '1'; fem       : analyse[8] := '2'; mascfem : analyse[8] := '3';
                    masc     : analyse[8] := '4'; mascneutre : analyse[8] := '5'; neutre  : analyse[8] := '6';
                    genrenul : analyse[8] := ' '  end;
                analyse[9] := ' '; analyse[10] := ' ';
        end;
adjpron : begin
                analyse[1] := '4';
                case noeud.catadjpron of
                    pers    : analyse[2] := '1'; pos : analyse[2] := '2'; refl  : analyse[2] := '3';
                    posrefl : analyse[2] := '4'; dem : analyse[2] := '5'; relat : analyse[2] := '6';
                    inter   : analyse[2] := '7'; ind : analyse[2] := '8'  end;
                case noeud.casadjpron of
                    nom    : case noeud.nbradjpron of sing : analyse[3] := 'A'; pluriel : analyse[3] := 'J' end;
                    voc    : case noeud.nbradjpron of sing : analyse[3] := 'B'; pluriel : analyse[3] := 'K' end;
                    acc    : case noeud.nbradjpron of sing : analyse[3] := 'C'; pluriel : analyse[3] := 'L' end;
                    gen    : case noeud.nbradjpron of sing : analyse[3] := 'D'; pluriel : analyse[3] := 'M' end;
                    datif  : case noeud.nbradjpron of sing : analyse[3] := 'E'; pluriel : analyse[3] := 'N' end;
                    abl    : case noeud.nbradjpron of sing : analyse[3] := 'F'; pluriel : analyse[3] := 'O' end;
                    loc    : case noeud.nbradjpron of sing : analyse[3] := 'G'; pluriel : analyse[3] := 'P' end;
                    indecl : analyse[3] := 'Z' end;
                case noeud.modesubadjpron of
                    indic   : analyse[4] := '1'; imp     : analyse[4] := '2'; subj     : analyse[4] := '3';
                    part    : analyse[4] := '4'; adjverb : analyse[4] := '5'; gerondif : analyse[4] := '6';
                    inf     : analyse[4] := '7'; supinum : analyse[4] := '8'; supinu   : analyse[4] := '9';
                    modenul : if ( (analyse[2] = '6') or (analyse[2] = '7') ) then analyse[4] := ' '
                                                                               else analyse[4] := '0' end;
                case noeud.tempssubadjpron of
                    pres     : analyse[5] := '1'; impft       : analyse[5] := '2'; futspl : analyse[5] := '3';
                    prft     : analyse[5] := '4'; plusqueprft : analyse[5] := '5'; futant : analyse[5] := '6';
                    fuisse   : analyse[5] := '7'; tempsnul : if ( (analyse[2] = '6') or (analyse[2] = '7') )
                                                  then analyse[5] := ' ' else analyse[5] := '0' end;
                analyse[6] := ' '; analyse[7] := ' ';
                case noeud.genreadjpron of
                    commun   : analyse[8] := '1'; fem       : analyse[8] := '2'; mascfem : analyse[8] := '3';
```

```
                masc     : analyse[8] := '4'; mascneutre : analyse[8] := '5'; neutre  : analyse[8] := '6';
                genrenul : analyse[8] := ' '  end;
            analyse[9] := ' '; analyse[10] := ' ';
        end;
v : begin
        if (noeud.lemme = 'SVM            ') then analyse[1] := 'E' else analyse[1] := '5';
        case noeud.conjv of
            c1 : case noeud.voixv of
                     actif : analyse[2] := '1'; passif : analyse[2] := 'A'; deponent : analyse[2] := 'J' end;
            c2 : case noeud.voixv of
                     actif    : analyse[2] := '2'; passif       : analyse[2] := 'B';
                     deponent : analyse[2] := 'K'; semideponent : analyse[2] := 'S' end;
            c3 : case noeud.voixv of
                     actif    : analyse[2] := '3'; passif       : analyse[2] := 'C';
                     deponent : analyse[2] := 'L'; semideponent : analyse[2] := 'T' end;
            c4 : case noeud.voixv of
                     actif : analyse[2] := '4'; passif : analyse[2] := 'D'; deponent : analyse[2] := 'M' end;
            c5 : case noeud.voixv of
                     actif : analyse[2] := '5'; passif : analyse[2] := 'E'; deponent : analyse[2] := 'N' end;
            c6 : case noeud.voixv of
                     actif : analyse[2] := '6'; passif : analyse[2] := 'F'; deponent : analyse[2] := 'O' end
        end;
        case noeud.modev of
            indic : begin
                      analyse[4] := '1';
                      case noeud.persv of
                           p1 : case noeud.nbr1v of sing : analyse[3]:='A'; pluriel : analyse[3]:='J' end;
                           p2 : case noeud.nbr1v of sing : analyse[3]:='B'; pluriel : analyse[3]:='K' end;
                           p3 : case noeud.nbr1v of sing : analyse[3]:='C'; pluriel : analyse[3]:='L' end  end
                    end;
            imp : begin
                    analyse[4] := '2';
                    case noeud.persv of
                         p1 : case noeud.nbr1v of sing : analyse[3]:='A'; pluriel : analyse[3]:='J' end;
                         p2 : case noeud.nbr1v of sing : analyse[3]:='B'; pluriel : analyse[3]:='K' end;
                         p3 : case noeud.nbr1v of sing : analyse[3]:='C'; pluriel : analyse[3]:='L' end  end
                  end;
            subj : begin
                     analyse[4] := '3';
                     case noeud.persv of
                          p1 : case noeud.nbr1v of sing : analyse[3]:='A'; pluriel : analyse[3]:='J' end;
                          p2 : case noeud.nbr1v of sing : analyse[3]:='B'; pluriel : analyse[3]:='K' end;
                          p3 : case noeud.nbr1v of sing : analyse[3]:='C'; pluriel : analyse[3]:='L' end  end
                   end;
            part : begin
                     analyse[4] := '4';
                     case noeud.casv of
                          nom : case noeud.nbr2v of sing : analyse[3] := 'A'; pluriel : analyse[3] := 'J' end;
                          voc : case noeud.nbr2v of sing : analyse[3] := 'B'; pluriel : analyse[3] := 'K' end;
                          acc : case noeud.nbr2v of sing : analyse[3] := 'C'; pluriel : analyse[3] := 'L' end;
                          gen : case noeud.nbr2v of sing : analyse[3] := 'D'; pluriel : analyse[3] := 'M' end;
                        datif : case noeud.nbr2v of sing : analyse[3] := 'E'; pluriel : analyse[3] := 'N' end;
                          abl : case noeud.nbr2v of sing : analyse[3] := 'F'; pluriel : analyse[3] := 'O' end;
                          loc : case noeud.nbr2v of sing : analyse[3] := 'G'; pluriel : analyse[3] := 'P' end;
                       indecl : analyse[3] := 'Z' end
                   end;
            adjverb : begin
                        analyse[4] := '5';
                        case noeud.casv of
                             nom : case noeud.nbr2v of sing : analyse[3]:='A'; pluriel : analyse[3]:='J' end;
                             voc : case noeud.nbr2v of sing : analyse[3]:='B'; pluriel : analyse[3]:='K' end;
                             acc : case noeud.nbr2v of sing : analyse[3]:='C'; pluriel : analyse[3]:='L' end;
                             gen : case noeud.nbr2v of sing : analyse[3]:='D'; pluriel : analyse[3]:='M' end;
                           datif : case noeud.nbr2v of sing : analyse[3]:='E'; pluriel : analyse[3]:='N' end;
                             abl : case noeud.nbr2v of sing : analyse[3]:='F'; pluriel : analyse[3]:='O' end;
```

```
                                loc : case noeud.nbr2v of sing : analyse[3]:='G'; pluriel : analyse[3]:='P' end;
                              indecl : analyse[3] := 'Z' end
                          end;
                gerondif : begin
                              analyse[4] := '6';
                              case noeud.casv of
                                   nom : case noeud.nbr2v of sing : analyse[3]:='A'; pluriel : analyse[3]:='J' end;
                                   voc : case noeud.nbr2v of sing : analyse[3]:='B'; pluriel : analyse[3]:='K' end;
                                   acc : case noeud.nbr2v of sing : analyse[3]:='C'; pluriel : analyse[3]:='L' end;
                                   gen : case noeud.nbr2v of sing : analyse[3]:='D'; pluriel : analyse[3]:='M' end;
                                 datif : case noeud.nbr2v of sing : analyse[3]:='E'; pluriel : analyse[3]:='N' end;
                                   abl : case noeud.nbr2v of sing : analyse[3]:='F'; pluriel : analyse[3]:='O' end;
                                   loc : case noeud.nbr2v of sing : analyse[3]:='G'; pluriel : analyse[3]:='P' end;
                                 indecl : analyse[3] := 'Z' end
                          end;
                inf : begin analyse[4] := '7'; analyse[3] := '0' end;
                supinum : begin analyse[4] := '8'; analyse[3] := '0' end;
                supinu : begin analyse[4] := '9'; analyse[3] := '0' end
           end;
           case noeud.tempsv of
                pres   : analyse[5] := '1'; impft       : analyse[5] := '2'; futspl : analyse[5] := '3';
                prft   : analyse[5] := '4'; plusqueprft : analyse[5] := '5'; futant : analyse[5] := '6';
                fuisse : analyse[5] := '7'; tempsnul    : analyse[5] := '0'  end;
           analyse[6] := ' '; analyse[7] := ' ';
           case noeud.genrev of
                commun   : analyse[8] := '1'; fem        : analyse[8] := '2'; mascfem : analyse[8] := '3';
                masc     : analyse[8] := '4'; mascneutre : analyse[8] := '5'; neutre  : analyse[8] := '6';
                genrenul : analyse[8] := ' ' end;
           analyse[9] := ' '; analyse[10] := ' '
    end;
adv : begin
           analyse[1] := '6';
           case noeud.catadv of
                rel    : analyse[2] := '6'; int  : analyse[2] := '7'; neg    : analyse[2] := '8';
                intneg : analyse[2] := '9'; comp : analyse[2] := ' '; superl : analyse[2] := '-';
                catnul : analyse[2] := '0'  end;
           analyse[3] := '0';
           case noeud.modesubadv of
                indic   : analyse[4] := '1'; imp     : analyse[4] := '2'; subj     : analyse[4] := '3';
                part    : analyse[4] := '4'; adjverb : analyse[4] := '5'; gerondif : analyse[4] := '6';
                inf     : analyse[4] := '7'; supinum : analyse[4] := '8'; supinu   : analyse[4] := '9';
                modenul : if ( (analyse[2] = '6') or (analyse[2] = '7') ) then analyse[4] := ' '
                                                                           else analyse[4] := '0' end;
           case noeud.tempssubadv of
                pres   : analyse[5] := '1'; impft       : analyse[5] := '2'; futspl : analyse[5] := '3';
                prft   : analyse[5] := '4'; plusqueprft : analyse[5] := '5'; futant : analyse[5] := '6';
                fuisse : analyse[5] := '7'; tempsnul    : if ( (analyse[2] = '6') or (analyse[2] = '7') )
                                                          then analyse[5] := ' ' else analyse[5] := '0' end;
           analyse[6] := ' '; analyse[7] := ' '; analyse[8] := ' '; analyse[9] := ' '; analyse[10] := ' '
    end;
prep : begin
           analyse[1] := '7';
           case noeud.typemecum of true : analyse[2] := '1'; false : analyse[2] := '0' end;
           case noeud.casprep of
                accusatif : analyse[3] := '3'; genitif : analyse[3] := '4'; ablatif : analyse[3] := '6' end;
           analyse[4] := '0'; analyse[5] := '0';
           analyse[6] := ' '; analyse[7] := ' '; analyse[8] := ' '; analyse[9] := ' '; analyse[10] := ' '
    end;
conj : begin
           analyse[1] := '8';
           case noeud.catconj of coord : analyse[2] := '1'; sub : analyse[2] := '2' end;
           analyse[3] := '0';
           case noeud.modesubconj of
                indic   : analyse[4] := '1'; imp     : analyse[4] := '2'; subj     : analyse[4] := '3';
                part    : analyse[4] := '4'; adjverb : analyse[4] := '5'; gerondif : analyse[4] := '6';
```

```
                        inf     : analyse[4] := '7'; supinum : analyse[4] := '8'; supinu   : analyse[4] := '9';
                        modenul : if (analyse[2] = '2') then analyse[4] := ' ' else analyse[4] := '0' end;
                    case noeud.tempssubconj of
                        pres   : analyse[5] := '1'; impft       : analyse[5] := '2'; futspl : analyse[5] := '3';
                        prft   : analyse[5] := '4'; plusqueprft : analyse[5] := '5'; futant : analyse[5] := '6';
                        fuisse : analyse[5] := '7'; tempsnul    : if (analyse[2] = '2') then analyse[5] := ' '
                                                                                        else analyse[5] := '0' end;
                    analyse[6] := ' '; analyse[7] := ' '; analyse[8] := ' '; analyse[9] := ' '; analyse[10] := ' '
                end;
            interj : begin
                        analyse[1] := '9'; analyse[2] := '0'; analyse[3] := '0'; analyse[4] := '0'; analyse[5] := '0';
                        analyse[6] := ' '; analyse[7] := ' '; analyse[8] := ' '; analyse[9] := ' '; analyse[10] := ' '
                     end
        end
end;

(*-------------------------------------------------------------------------*)

procedure TRT_TRAD;

var lnatemp, noeudfilscour : listnoeudaa;
    firstltflcour, ltflcour, newltfl, firstltfl, newltflcour, ltflcourprec : listtradfranc;
    analmorphol : str10;
    ch : char;
    stop, ok, accept, trouve : boolean;
    longreelle, k, position : integer;
    motaecrire : str24;
    mcpcour : lcontphr;

label FINTRTTRAD, AFFICH;

begin
     finanbool := false;
(* recherche des traductions des mots s{lectionn{s selon une certaine A.M. *)
(* et cr{ation d'une liste pour affichage.                                  *)
     if (accesfamml = false) then begin assign (famml, 'b:fmotlat.am'); reset (famml); accesfamml := true end;
     ltflcour := nil; lnatemp := lnacour;
     while (lnatemp <> nil) do
     begin
          case lnatemp^.noeud.typenoeud of
          n2 : begin
                    TRSF (lnatemp^.noeud, analmorphol);
                    FIND_INDEX (lnatemp^.noeudpere^.noeud.mot1, position);
                    seek (famml, position);
                    trouve := false;
                    while (trouve = false) do
                    begin
                         read (famml, lignefamml);
                         if ( (lignefamml.catgram = analmorphol[1])
                          and (lignefamml.sscatgram_dgr_vx = analmorphol[2])
                          and (lignefamml.cas_pers_nbr = analmorphol[3])
                          and (lignefamml.mode = analmorphol[4])
                          and (lignefamml.temps = analmorphol[5])
                          and (lignefamml.fonction = analmorphol[6])
                          and (lignefamml.emplois = analmorphol[7])
                          and (lignefamml.genre = analmorphol[8])
                          and (lignefamml.codesubord[1] = analmorphol[9])
                          and (lignefamml.codesubord[2] = analmorphol[10]) )
                         then begin
                                   trouve := true; new (newltfl);
                                   newltfl^.motlatin := lnatemp^.noeudpere^.noeud.mot1;
                                   newltfl^.tradfranc := lignefamml.tradf;
                                   newltfl^.suivant_tf := nil;
                                   if (ltflcour = nil)
                                   then begin firstltflcour := newltfl; ltflcour := newltfl end
```

```
                                    else begin ltflcour^.suivant_tf := newltfl; ltflcour := ltflcour^.suivant_tf end
                               end
                          end;
                          lnatemp := lnatemp^.noeudpere^.noeudpere
                     end;
               n3 : lnatemp := lnatemp^.noeudpere^.noeudpere^.noeudpere
               end
     end;
(* classement de la liste selon l'ordre des mots de la phrase *)
     mcpcour := firstmcp; firstltfl := nil; newltflcour := nil;
     while (mcpcour <> nil) do
     begin
          ltflcour := firstltflcour; ltflcourprec := nil;
          while (ltflcour^.motlatin <> mcpcour^.mot) do
          begin  ltflcour := ltflcour^.suivant_tf;
                 if (ltflcourprec = nil) then ltflcourprec := firstltflcour
                                         else ltflcourprec := ltflcourprec^.suivant_tf
          end;
          new (newltfl); newltfl^.motlatin := ltflcour^.motlatin; newltfl^.tradfranc := ltflcour^.tradfranc;
          newltfl^.suivant_tf := nil;
          if (newltflcour = nil)
          then begin firstltfl := newltfl; newltflcour := newltfl end
          else begin newltflcour^.suivant_tf := newltfl; newltflcour := newltflcour^.suivant_tf end;
          if (ltflcourprec = nil) then firstltflcour := ltflcour^.suivant_tf
                                  else ltflcourprec^.suivant_tf := ltflcour^.suivant_tf;
          mcpcour := mcpcour^.suivant
     end;
     (* pas de cr{ation de noeud : uniquement affichage traduction *)
AFFICH :
     accept := false;
     while (accept = false) do
     begin
          clrscr; writeln ('Voici une traduction de la phrase ci-dessus, selon ton analyse :'); writeln;
          ltflcour := firstltfl;
          while (ltflcour <> nil) do
          begin
               if (wherey > 8) then begin
                                         gotoxy (35, 10);
                                         write ('Tape une touche pour voir la page suivante ! ');
                                         repeat until keypressed; ch := readkey; clrscr
                                    end;
               stop := false; k := 24;
               while ( (stop = false) and (k > 0) ) do
                    if (ltflcour^.tradfranc[k] <> ' ') then stop := true else k := k - 1;
               longreelle := k;
               motaecrire := copy (ltflcour^.tradfranc, 1, longreelle);
               if ((80 - wherex) < longreelle) then writeln;
               write (motaecrire);
               if (wherex = 1) then gotoxy (80, wherey - 1)
                               else begin write (' '); if (wherex = 1) then gotoxy (80, wherey - 1) end;
               ltflcour := ltflcour^.suivant_tf
          end;
          gotoxy (1, 10);
          write ('Veux-tu des explications @ propos de cette ''traduction'' ? ''O''-''N'' : _');
          gotoxy (wherex - 1, wherey); ok := false;
          while (ok = false) do
          begin
               if keypressed then begin
                                       ch := readkey;
                                       case ch of
                                            'O', 'o', 'N', 'n' : begin write (ch); ok := true end;
                                            else                 begin sound (220); delay (200); nosound end
                                       end
                                  end
          end;
```

```
          if ( (ch = 'N') or (ch = 'n') )
          then accept := true
          else begin
                    ltflcour := firstltfl;
                    while (ltflcour <> nil) do
                    begin
                         gotoxy (1, 12); delline; stop := false; k := 24;
                         while ( (stop = false) and (k > 0) ) do
                              if (ltflcour^.tradfranc[k] <> ' ') then stop := true else k := k - 1;
                         longreelle := k;
                         motaecrire := copy (ltflcour^.tradfranc, 1, longreelle);
                         writeln ('Traduction de ', ltflcour^.motlatin, ' : ', motaecrire);
                         write ('Tape une touche pour voir la suite');
                         repeat until keypressed; ch := readkey;
                         ltflcour := ltflcour^.suivant_tf
                    end
               end
     end;
     noeudfilscour := lnacour^.noeudfils;
     gotoxy (1, 10); delline; delline; delline;
     if (noeudfilscour^.noeud.accept_comment = true)
     then begin
            gotoxy (1, 10);
            if (noeudfilscour^.noeud.comment_specif <> '')
            then write (noeudfilscour^.noeud.comment_specif) else write (noeudfilscour^.noeud.comment_syst)
          end;
     if (noeudfilscour^.noeud.ind_brk = break)
     then begin
            gotoxy (1, 12);
            writeln ('Traduction insens{e ! Tu as commis une ERREUR IMPARDONNABLE dans ton analyse ! ');
            write ('Tape une touche pour corriger ton analyse ou ESC !');
            repeat until keypressed; ch := readkey;
            if (ch = #27) then begin finanbool := true; goto FINTRTTRAD end;
            noeudfilscour := noeudfilscour^.noeudpere; lnacour := lnacour^.noeudpere; goto FINTRTTRAD
          end
     else begin
             gotoxy (1, 12); finanbool := true; writeln ('F{licitations !');
             writeln ('Tu es parvenu @ analyser correctement cette phrase et la traduction est sens{e.');
             writeln; write ('Tape une touche pour choisir une autre phrase @ analyser !');
             repeat until keypressed; ch := readkey
          end;
FINTRTTRAD :
     write ('')
end;


End.
```

```
Program LATELEVE;

uses ('$U b:globalel ') GLOBALEL, CRT, DOS, ('$U b:cpltel ') CPLTEL,
     ('$U b:le_unit ') LE_UNIT, ('$U b:trts ') TRTS;

const colortxt  = 14;     (* yellow *)
      colorfond = 0;      (* black  *)

var  out, gotoend, oke : boolean;
     ch                : char;
     namefct, namefdt  : str14;
     iocode            : integer;


label DEBUTPGR;


procedure COPY_FDSC_LDSC ( var namefdt : str14; var firstld : list_descr);

(* specifications : copie le fichier d{crivant un texte latin dans une liste *)
(*                  chai^n{e plus facilement manipulable.                    *)

var ldcour, newld : list_descr;

begin
      assign (fdescrtxt, namefdt); reset (fdescrtxt); seek (fdescrtxt, 0);
      ldcour := nil;
      while ( not eof(fdescrtxt) ) do
      begin
            read (fdescrtxt, lignefdt);
            new (newld);
            if (lignefdt.typedescr = txtdescr )
            then with newld^.contenu do
                 begin
                       typedescr := lignefdt.typedescr;
                       titre_txt := lignefdt.titre_txt;
                       etat_prep_txt := lignefdt.etat_prep_txt;
                       res_trt_aut := lignefdt.res_trt_aut;
                       comment_txt := lignefdt.comment_txt;
                       nomfichcontxt := lignefdt.nomfichcontxt
                 end
            else with newld^.contenu do
                 begin
                       typedescr := lignefdt.typedescr;
                       numero_phrase := lignefdt.numero_phrase;
                       etat_prep_phrase := lignefdt.etat_prep_phrase;
                       res_trt_aut_phrase := lignefdt.res_trt_aut_phrase;
                       comment_phrase := lignefdt.comment_phrase;
                       nomfichanal := lignefdt.nomfichanal
                 end;
            newld^.suivant := nil;
            if ( ldcour = nil )
            then begin ldcour := newld; firstld := newld end
            else begin ldcour^.suivant := newld; ldcour := ldcour^.suivant end
      end;
      close (fdescrtxt)
end;

(*************************************************************************)

procedure TRSF_PHR_L (var namefct : str14; nphrase : integer; var fmcp : lcontphr);

(* sp{cifications : TRSF_PHR_L re\oit 'nphrase', le num{ro d'une phrase     *)
(*                  appartenant au texte latin m{moris{ dans le fichier     *)
(*                  portant le nom 'namefct', et recopie cette phrase dans  *)
```

```
(*                une liste chaînée plus facilement manipulable.          *)
(*                fmcp contiendra un pointeur pointant vers le premier    *)
(*                maillon de cette liste chaînée.                         *)

const  motb = '                    ';
var j, k, min : integer;
    finphrase, finligne : boolean;
    mcpcour, newmcp : lcontphr;
    bigmot : string[40];

begin
     assign (fconttxt, namefct); reset (fconttxt);
     j := 0;
     while ( (not eof(fconttxt)) and (j < nphrase - 1) ) do
     begin
          readln (fconttxt, lignefconttxt);
          if ( lignefconttxt[length(lignefconttxt)] = '.' ) then j := j + 1;
     end;
     finphrase := false; mcpcour := nil;
     while ( (not eof(fconttxt)) and (finphrase = false) ) do
     begin
          readln (fconttxt, lignefconttxt);
          k := 0; finligne := false; min := 1;
          while ( (k < 80) and (finligne = false) ) do
          begin
               repeat k := k + 1 until ( (k = 80) or (lignefconttxt[k] = ' ') or (lignefconttxt[k] = '.') );
               new (newmcp);
               if ( (k = 80) and (lignefconttxt[k] <> ' ') and (lignefconttxt[k] <> '.') )
               then begin
                         newmcp^.mot := copy (lignefconttxt, min, k - min + 1);
                         bigmot := concat (newmcp^.mot, motb);
                         newmcp^.mot := copy (bigmot, 1, 20)
                    end;
               if ( (k = 80) and ( (lignefconttxt[k] = ' ') or (lignefconttxt[k] = '.') ) )
               then begin
                         newmcp^.mot := copy (lignefconttxt, min, k - min);
                         bigmot := concat (newmcp^.mot, motb);
                         newmcp^.mot := copy (bigmot, 1, 20)
                    end;
               if (k < 80)
               then begin
                         newmcp^.mot := copy (lignefconttxt, min, k - min);
                         bigmot := concat (newmcp^.mot, motb);
                         newmcp^.mot := copy (bigmot, 1, 20);
                         if ( (lignefconttxt[k] = '.') or (lignefconttxt[k+1] = ' ') )
                         then finligne := true
                         else min := k + 1
                    end;
               newmcp^.libre := true; newmcp^.suivant := nil;
               if (mcpcour = nil)
               then begin mcpcour := newmcp; fmcp := newmcp end
               else begin mcpcour^.suivant := newmcp; mcpcour := mcpcour^.suivant end
          end;
          if (lignefconttxt[length(lignefconttxt)] = '.') then finphrase := true
     end;
     close (fconttxt)
end;

(*************************************************************************)

procedure COPY_FA_LA (var nomfichanal : str14; var firstlna : listnoeudaa);

var newlna, lnacour, oldlna, fauxnoeudfrere : listnoeudaa;
    oldfna : fnoeudaa;
```

```
begin
      new (fauxnoeudfrere);
      fauxnoeudfrere^.noeudfils := nil; fauxnoeudfrere^.noeudfrere := nil; fauxnoeudfrere^.noeudpere := nil;
      with fauxnoeudfrere^.noeud do
      begin
           ind_brk := break;
           appart_analyse_correct := false;
           comment_syst := '';
           comment_specif := '';
           accept_comment := false;
           typenoeud := n1;
           fonction := sujet;
           mot1 := 'faux noeud frere !'
      end;
      assign (fna, nomfichanal); reset (fna);
      lnacour := nil;
      while ( not eof(fna) ) do
      begin
           read (fna, fnaligne);
           new (newlna);
           newlna^.noeud := fnaligne.noeud;
           if (fnaligne.noeudfils = false) then newlna^.noeudfils := nil;
           if (fnaligne.noeudfrere = false) then newlna^.noeudfrere := nil
                                            else newlna^.noeudfrere := fauxnoeudfrere;
           if (lnacour = nil)
           then begin lnacour := newlna; newlna^.noeudpere := nil; firstlna := newlna end
           else if (oldfna.noeudfils = true)
                then begin oldlna^.noeudfils := newlna; newlna^.noeudpere := oldlna; lnacour := newlna end
                else begin
                          while (oldlna^.noeudfrere <> fauxnoeudfrere)
                          do oldlna := oldlna^.noeudpere;
                          oldlna^.noeudfrere := newlna; newlna^.noeudpere := oldlna^.noeudpere; lnacour := newlna
                     end;
           oldfna := fnaligne; oldlna := newlna
      end;
      close (fna)
end;

(*************************************************************************)

procedure COPY_FAEL_LAEL (var nomfichanalel : str14; var firstlnael : listnoeudael);

var newlnael, lnacourel, oldlnael : listnoeudael;
    oldfnael : fnoeudael;

begin
      assign (fnael, nomfichanalel); reset (fnael);
      lnacourel := nil;
      while ( not eof(fnael) ) do
      begin
           read (fnael, fnaelligne);
           new (newlnael); newlnael^.noeudel := fnaelligne.noeudel;
           if (fnaelligne.noeudfilsel = false) then newlnael^.noeudfilsel := nil;
           if (lnacourel = nil)
           then begin lnacourel := newlnael; newlnael^.noeudpereel := nil; firstlnael := newlnael end
           else begin oldlnael^.noeudfilsel := newlnael; newlnael^.noeudpereel := oldlnael; lnacourel := newlnael end;
           oldfnael := fnaelligne; oldlnael := newlnael
      end;
      close (fnael)
end;


(*************************************************************************)

procedure COPY_LAEL_FAEL (var nomfichanalel : str14; var firstlnael : listnoeudael);
```

```
var lnacourel : listnoeudael;

begin
     lnacourel := firstlnael;
     assign (fnael, nomfichanalel); rewrite (fnael);
     while (lnacourel <> nil) do
     begin
          if (lnacourel^.noeudfilsel <> nil) then fnaelligne.noeudfilsel := true
                                             else fnaelligne.noeudfilsel := false;
          fnaelligne.noeudel := lnacourel^.noeudel;
          write (fnael, fnaelligne);
          lnacourel := lnacourel^.noeudfilsel
     end;
     close (fnael)
end;

{*************************************************************************}

function VALID_TXT (var nametxt : str14) : boolean;

{ specifications : si 'b:nametxt.fct' correspond au titre d'un texte latin existant }
{                  alors VALID_TXT prend la valeur vrai                             }
{                  sinon VALID_TXT prend la valeur faux                             }

begin
     nametxt := concat ('b:', nametxt, '.fct' );
{$I-}
     assign (fconttxt, nametxt); reset (fconttxt);
     if (ioresult = 0) then begin valid_txt := true; close (fconttxt) end
                       else valid_txt := false
{$I+}
end;

{*************************************************************************}

procedure LIST_TXT_LAT;

{ sp{cifications : LIST_TXT_LAT fournit la liste des fichiers '*.fct' se    }
{                  trouvant sur le drive 'b';                               }
{                  ces fichiers contiennent chacun un texte latin.          }


var dirinfo        : searchrec;
    ymax, k, iocode : integer;
    ch             : char;
    titre          : str30;
    namefdt        : str14;

begin
     clrscr; gotoxy (7,2);
     writeln ('LISTE DES FICHIERS CONTENANT DES TEXTES LATINS SUR LE DRIVE ''B'' :');
     writeln; writeln;
     ymax := 23;
     findfirst ('b:*.fct', anyfile, dirinfo);
     while (doserror = 0) do
     begin
          if (wherey > ymax)
          then begin
                    gotoxy (35,25);
                    write ('Tape une touche pour voir la page suivante');
                    repeat until keypressed; ch := readkey; clrscr;
                    gotoxy (7, 2);
                    writeln ('LISTE DES FICHIERS CONTENANT DES TEXTES LATINS SUR LE DRIVE ''B'' :');
                    writeln; writeln
```

```
            end;
          write (dirinfo.name);
          k := length(dirinfo.name) - 4;
          namefdt := copy (dirinfo.name, 1, k);
          namefdt := concat ('b:', namefdt, '.dsc');
{$I-}
          assign (fdescrtxt, namefdt); reset (fdescrtxt);
          iocode := ioresult;
          if (iocode <> 0)
          then titre := ''
          else begin read (fdescrtxt, lignefdt); titre := lignefdt.titre_txt; close (fdescrtxt) end;
{$I+}
          gotoxy (25, wherey);
          writeln (titre);
          findnext (dirinfo)
    end;
    gotoxy (35,25); write ('Tape une touche pour retourner au menu');
    repeat until keypressed; ch := readkey
end;

{**************************************************************************}

procedure ANALYSE_ELEVE (var namefct : str14);

{ sp(cifications : ANALYSE_ELEVE permet de visualiser l'analyse existante  }
{                  des phrases contenues dans le texte 'namefct' et de les }
{                  compl(ter.                                              }


var
    k, iocode, ymin, ymax, nbrephrase, i : integer;
    namefdt,nomfna, nomfnael : str14;
    titretxt, titre : str60;
    finanbool, right, ok, oke, existfnael, accesfamml, stop : boolean;
    remontev, remontegns, remontecod, remontecplt  : boolean;
    ch    : char;
    comment : str80;
    ldcour, firstld : list_descr;
    stri : string[3];
    firstmcp, mcpcour : lcontphr;
    firstlna, lnacour : listnoeudaa;
    firstlnael, lnacourel : listnoeudael;
    trt, trtsuivant : traitement;

label FIN, FINAN, CHOIXPHR ;


Begin
    clrscr;
    accesfamml := false;
    k := length(namefct) - 4;
    namefdt := copy (namefct, 1, k);
    namefdt := concat (namefdt, '.dsc');
{$I-}
    assign (fdescrtxt, namefdt); reset (fdescrtxt);
    iocode := ioresult;
    if (iocode <> 0)
    then begin
              writeln (namefdt, ' n''existe pas ! Tape une touche pour revenir au menu !');
              repeat until keypressed; ch := readkey; goto FIN
         end
    else begin
              nbrephrase := filesize (fdescrtxt) - 1;
              read (fdescrtxt, lignefdt);
              titretxt := lignefdt.titre_txt;
```

```
                comment := lignefdt.comment_txt;
                close (fdescrtxt);
                COPY_FDSC_LDSC (namefdt, firstld)
           end;
(*$I+*)
     ymin := 1; ymax := 19;
     VISUTXTPHR (namefct, 0, titretxt, ymin, ymax, right);
     gotoxy (1, 21);
     write ('Veux-tu voir s''afficher le commentaire de ce texte ? ''O''-''N'' : _');
     gotoxy (wherex - 1, wherey); ok := false;
     while (ok = false) do
     begin
       if keypressed then begin
                          ch := readkey;
                          case ch of
                               'O', 'o' : begin ok := true; writeln (ch);
                                                writeln; write (firstld^.contenu.comment_txt) end;
                               'N', 'n' : begin ok := true; writeln (ch) end;
                               else       begin sound (220); delay (200); nosound end
                          end
                        end
     end;
     gotoxy (40, 25);
     write ('Tape une touche pour continuer !');
     repeat until keypressed; ch := readkey;
CHOIXPHR :
     window (1, 1, 80, 25); clrscr;
     ok := false;
     i := 1;
     while (ok = false) do
     begin
          str (i, stri);
          titre := concat ('SELECTION DE LA PHRASE A ANALYSER : phrase n[', stri);
          VISUTXTPHR (namefct, i, titre, 5, 19, right);
          gotoxy (1, 21);
          write ('Veux-tu analyser cette phrase ? ''O''ui, ''N''on ou ''ESC'' : ');
          oke := false;
          while (oke = false) do
          begin
               if keypressed
               then begin
                         ch := readkey;
                         case ch of
                              'O', 'o' : begin write (ch); oke := true; ok := true end;
                              'N', 'n' : begin
                                                write (ch);
                                                oke := true;
                                                i := i + 1;
                                                if (i > nbrephrase) then i := 1
                                         end;
                              #27      : goto FIN;
                              else       begin sound (220); delay (200); nosound end
                         end
                    end
          end
     end;
     delline;
     ldcour := firstld;
     k := 1;
     while (k <= i) do
     begin
          ldcour := ldcour^.suivant;
          k := k + 1
     end;
     gotoxy (1, 21);
```

```
    write ('Veux-tu voir s''afficher le commentaire de cette phrase ? ''O''-''N'' : _');
    gotoxy (wherex - 1, wherey); ok := false;
    while (ok = false) do
    begin
        if keypressed then begin
                                ch := readkey;
                                case ch of
                                    'O', 'o' : begin
                                                   ok := true;
                                                   writeln (ch);
                                                   writeln;
                                                   write (ldcour^.contenu.comment_phrase)
                                               end;
                                    'N', 'n' : begin ok := true; writeln (ch) end;
                                    else       begin sound (220); delay (200); nosound end
                                end
                            end
    end;

(* analyse de la phrase i si possible *)
    if (ldcour^.contenu.etat_prep_phrase <> e3)
    then begin
             gotoxy (1, 25);
             write ('Phrase non pre^te @ e^tre analys{e, tape une touche pour en choisir une autre');
             repeat until keypressed;
             ch := readkey;
             goto CHOIXPHR
         end;
    reset (fdescrtxt);
    seek (fdescrtxt, i);
    read (fdescrtxt, lignefdt);
    (*$I-*)
    assign (fna, lignefdt.nomfichanal);
    reset (fna);
    iocode := ioresult;
    if (iocode <> 0)
    then begin
             gotoxy (1, 25);
             write ('Analyse du professeur inexistante, tape une touche pour choisir une autre phrase');
             repeat until keypressed;
             ch := readkey;
             goto CHOIXPHR
         end
    else begin
             close (fna);
             COPY_FA_LA (lignefdt.nomfichanal, firstlna);
             gotoxy (40, 25);
             write ('Tape une touche pour continuer !');
             repeat until keypressed;
             ch := readkey
         end;
    (*$I+*)
    close (fdescrtxt);

    (* phrase analysable et analyse du professeur existante : analyse de l'{l}ve *)
    titre := firstld^.contenu.titre_txt;
    str (i, stri);
    titre := concat (titre, ' phrase n[', stri);
    VISUTXTPHR (namefct, i, titre, 1, 10, right);
    window (1, 11, 80, 25);
    TRSF_PHR_L (namefct, i, firstmcp);

(* dans le cas o¦ l'analyse existe d{j@, il faut la reprendre !          *)
    existfnael := false;
    (*$I-*)
```

```
k := length (namefct) - 4;
nomfnael := copy (namefct, 1, k);
nomfnael := concat (nomfnael, '.', stri[1], 'el');
assign (fnael, nomfnael);
reset (fnael);
iocode := ioresult;
if (iocode = 0)
then begin
          close (fnael);
          existfnael := true;
          COPY_FAEL_LAEL (nomfnael, firstlnael)
     end;
{$I+}

trt := trtv;
stop := false;
remontev := false;

while (stop = false) do
begin
     case trt of
          trtv :  { traitement du verbe principal }
                begin
                  TRT_VP (existfnael,firstmcp,lnacour,firstlna,lnacourel,firstlnael,finanbool,
                         remontev,namefct,titretxt,firstld,ldcour);
                  if (finanbool = true) then stop := true
                                        else begin
                                               trt := trtgns;
                                               remontegns := false
                                             end
                end;
          trtgns :  { traitement GNsujet : choix sujet, analyses morphologiques et d{coupe      }
                    { syntaxique suivie de la s{lection et des analyses de chaque mot.          }
                    begin
                      TRT_GN (existfnael,firstmcp,lnacour,lnacourel,finanbool,nom,remontegns,
                             trtsuivant,namefct,titretxt,firstld,ldcour);
                      if (finanbool = true) then stop := true
                                            else begin
                                                   if (trtsuivant = trtnul)
                                                   then begin
                                                          if (lnacourel^.noeudfilsel = nil)
                                                          then existfnael := false else existfnael := true;
                                                          trt := trtgncod;
                                                          remontecod := false
                                                        end
                                                   else begin
                                                          trt := trtsuivant;  { trtv }
                                                          remontev := true
                                                        end
                                                 end
                    end;
          trtgncod : { traitement GNcod : choix cod, analyses morphologiques et d{coupe }
                     { syntaxique suivie de la s{lection et des analyses de chaque mot. }
                     begin
                       TRT_GN (existfnael,firstmcp,lnacour,lnacourel,finanbool,acc,remontecod,
                              trtsuivant,namefct,titretxt,firstld,ldcour);
                       if (finanbool = true) then stop := true
                                             else begin
                                                    if (trtsuivant = trtnul)
                                                    then begin
                                                           if (lnacourel^.noeudfilsel = nil)
                                                           then existfnael := false else existfnael := true;
                                                           trt := trtgncplt;
                                                           remontecplt := false
                                                         end
```

```
                                                    else begin
                                                           trt := trtsuivant; (* trtgns *)
                                                           remontegns := true
                                                         end
                                                  end
                        end;
              trtgncplt : (* traitement GNcompl(ment : choix cplt, analyses morphologiques et d(coupe *)
                          (* syntaxique suivie de la s(lection et des analyses de chaque mot.        *)
                          begin
                            TRT_GN (existfnael,firstmcp,lnacour,lnacourel,finanbool,abl,remontecplt,
                                    trtsuivant,namefct,titretxt,firstld,ldcour);
                            if (finanbool = true) then stop := true
                                                  else begin
                                                       if (trtsuivant = trtnul)
                                                       then begin
                                                              if (lnacourel^.noeudfilsel = nil)
                                                              then existfnael := false else existfnael := true;
                                                              trt := traduct
                                                            end
                                                       else begin
                                                              trt := trtsuivant;  (* trtgncod *)
                                                              remontecod := true
                                                            end
                                                       end
                          end;
                traduct :  (* traitement traductions *)
                          begin
                            TRT_TRAD (firstmcp,accesfamml,lnacour,finanbool);
                            if (finanbool = true) then stop := true
                                                  else begin
                                                         trt := trtgncplt;
                                                         remontecplt := true;
                                                         existfnael := true;
                                                         lnacourel := lnacourel^.noeudpereel;
                                                       end
                        end
          end
    end;

(* recopie de la liste chai^n(e des noeuds d'analyse dans le fichier          *)
(* et retour @ l'endroit o} l'on peut choisir une autre phrase @ analyser     *)
FINAN:
    COPY_LAEL_FAEL (nomfnael, firstlnael);
    goto CHOIXPHR;

FIN :
    if (accesfamml = true) then close (famml)
end;


(************************** d(but program prof_lat ******************************)

BEGIN
DEBUTPGR :
    textcolor (colortxt);
    textbackground (colorfond);
    window (1,1,80,25);
    clrscr;
    gotoxy (8,1);
    writeln ('DIDACTICIEL D''AIDE A L''ANALYSE ET LA TRADUCTION DE TEXTES LATINS :');
    gotoxy (8,2);
    writeln ('-------------------------------------------------------------');
    gotoxy (35,4);
    writeln( 'LECONS');
    gotoxy (35,5);
```

```
writeln ('------');
gotoxy (7,9);
writeln('1- LISTE TEXTES LATINS');
writeln;
writeln ('      2- VISUALISATION TEXTE LATIN');
writeln;
writeln ('      3- ANALYSE ET TRADUCTION TEXTE LATIN');
writeln;
writeln ('      4- FIN');
gotoxy (18,20);
write ('TAPE LE NUMERO CORRESPONDANT A TON CHOIX : _');
gotoxy (wherex -1, wherey);
out := false;
while ( out = false ) do
begin
     if keypressed
     then begin
               ch := readkey;
               case ch of
                    '1' : begin
                              write (ch);
                              LIST_TXT_LAT;
                              goto DEBUTPGR
                         end;
                    '2' : begin
                              write (ch);
                              clrscr;
                              gotoend := false;
                              oke := false;
                              repeat
                                   write ('Nom du texte latin @ visualiser ou RETURN : b: _______.fct');
                                   gotoxy (48, wherey);
                                   readln (namefct);
                                   if namefct = ''
                                   then gotoend := true
                                   else if (VALID_TXT (namefct) = false)
                                        then writeln (namefct, ' n''existe pas !')
                                        else oke := true
                              until ( (oke = true) or (gotoend = true) );
                              if (gotoend <> true) then VISU_TXT_LAT(namefct);
                              goto DEBUTPGR
                         end;
                    '3' : begin
                              write (ch);
                              clrscr;
                              gotoend := false;
                              oke := false;
                              repeat
                                   write ('Nom du texte latin @ analyser ou RETURN : ');
                                   write ('b: _______.fct');
                                   gotoxy (46, wherey);
                                   readln (namefct);
                                   if namefct = ''
                                   then gotoend := true
                                   else if (VALID_TXT (namefct) = false)
                                        then writeln (namefct, ' n''existe pas !')
                                        else oke := true
                              until ( (oke = true) or (gotoend = true) );
                              if (gotoend <> true) then ANALYSE_ELEVE(namefct);
                              goto DEBUTPGR
                         end;
                    '4' : begin
                              write (ch);
                              out := true
                         end;
```

```
                    else  begin
                                sound (220);
                                delay (200);
                                nosound
                          end
                end
          end
    end
END.
```